

AAMAS 2011

The 10th International Conference on Autonomous Agents and Multiagent Systems

May 2–6, 2011 • Taipei, Taiwan

Proceedings
Volume I



IFAAMAS

International Foundation for Autonomous Agents and Multiagent Systems

www.ifaamas.org

Copyright © 2011 by the International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS). Permissions to make digital or hard copies of portions of this work for personal or classroom use is granted without fee provided that the copies are not made or distributed for profit or commercial advantage and that the copies bear the full citation on the first page. Copyrights for components of this work owned by others than IFAAMAS must be honoured. Abstracting with credit is permitted.

To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or fee. Request permission to republish from the IFAAMAS board of directors via info@ifaamas.org

ISBN-10: 0-9826571-5-3

ISBN-13: 978-0-9826571-5-7

Introduction

The Autonomous Agents and MultiAgent Systems (AAMAS) conference series brings together researchers from around the world to share the latest advances in the field. It provides a marquee, high-profile forum for research in the theory and practice of autonomous agents and multiagent systems. AAMAS 2002, the first of the series, was held in Bologna, followed by Melbourne (2003), New York (2004), Utrecht (2005), Hakodate (2006), Honolulu (2007), Estoril (2008), Budapest (2009) and Toronto (2010). You are now about to enter the proceedings of AAMAS 2011, held in Taipei, Taiwan, as AAMAS celebrates its 10th anniversary as the successful merger of three related events that had run for some years previously.

In addition to the general track for the AAMAS 2011 conference, submissions were invited to three special tracks: a Robotics track, a Virtual Agents track and an Innovative Applications track. The aims of these special tracks were to give researchers from these areas a strong focus, to provide a forum for discussion and debate within the encompassing structure of AAMAS, and to ensure that the impact of both theoretical contributions and innovative applications were recognized. Each track was chaired by a leader in the field: Maria Gini for the robotics track, James Lester for the virtual agents track, and Peter McBurney for the innovative applications track. The special track chairs provided critical input to selection of Program Committee (PC) and Senior Program Committee (SPC) members, and to the reviewer allocation and the review process itself. The final decisions concerning acceptance of papers were taken by the AAMAS 2011 Program Co-chairs in discussion with, and in full agreement with the special track chairs.

Only full paper submissions were solicited for AAMAS 2011. The general, robotics, virtual agents, and innovative applications tracks received 452, 31, 51, and 41 submissions respectively, for a total of 575 submissions.

After a thorough and exciting review process, 126 papers were selected for publication as Full Papers each of which was allocated 8 pages in the proceedings and allocated 20 minutes in the Program for oral presentation. Another 123 papers were selected as Extended Abstracts and allocated 2 pages each in the proceedings. Both Full Papers and Extended Abstracts are presented as posters during the conference.

Of the submissions, more than half (338) have a student as first author, which indicates an exciting future for the field. Representation under all submissions of topics (measured by first keyword) was broad, with top counts in areas such as teamwork, coalition formation, and coordination (31), distributed problem solving (30), game theory (30), planning (26), multiagent learning (24), and trust, reliability and reputation (17).

We thank the PC and SPC members of AAMAS 2011 for their thoughtful reviews and extensive discussions. We thank Maria Gini, James Lester and Peter McBurney for making the Robotics, the Virtual Agents and the Innovative Applications tracks a success. We thank Michael Rovatsos for putting together the proceedings. Finally, we thank David Shield for his patience and support regarding Confmaster during every stage between the submission process and the actual AAMAS 2011 event. The Program represents the intellectual motivation for researchers to come together at the Conference, but the success of the event is dependent on the many other elements that make up the week especially the tutorials, workshops, and doctoral consortium. We thank all members of the Conference Organising Committee for their dedication, enthusiasm, and attention to detail, and wish to particularly thank Von-Wun Soo as Chair of the Local Organising Committee for his contributions.

*Kagan Tumer and Pinar Yolum,
AAMAS 2011 Program Co-Chairs*

*Peter Stone and Liz Sonenberg,
AAMAS 2011 General Co-Chairs*



Organizing Committee

General Chairs

Liz Sonenberg (The University of Melbourne, Australia)
Peter Stone (The University of Texas at Austin, USA)

Program Co-Chairs

Kagan Tumer (Oregon State University, USA)
Pinar Yolum (Bogazici University, Turkey)

Robotics Track Chair

Maria Gini (University of Minnesota, USA)

Virtual Agents Track Chair

James Lester (North Carolina State University, USA)

Innovative Applications Chair

Peter McBurney (University of Liverpool, UK)

Local Arrangements Chair

Von-Wun Soo (National Tsing Hua University, Taiwan)

Local Arrangements Committee

Tzung-Pei Hong (National University of Kaohsiung, Taiwan)
Churn-Jung Liao (Academia Sinica, Taiwan)
Chao-Lin Liu (National Chengchi University, Taiwan)
Soe-Tsyr Yuan (National Chengchi University, Taiwan)

Finance Chair

Nancy Reed (University of Hawaii, USA)

Publicity Chair

Iyad Rahwan (Masdar Institute, UAE)

Publications Chair

Michael Rovatsos (The University of Edinburgh, UK)

Tutorials Chair

Vincent Conitzer (Duke University, USA)

Workshops Chair

Frank Dignum (Universiteit Utrecht, Netherlands)

Exhibitions Chair

Sonia Chernova (Worcester Polytechnic Institute, USA)

Demonstrations Chair

Elizabeth Sklar (City University of New York, USA)

Scholarships Co-Chairs

Matthew E. Taylor (Lafayette College, USA)
Michael Winikoff (University of Otago, New Zealand)

Doctoral Consortium Co-Chairs

Kobi Gal (Ben-Gurion University of the Negev, Israel)
Adrian Pearce (The University of Melbourne, Australia)

Sponsorship Co-Chairs

Sherief Abdallah (British University in Dubai, UAE)
Jane Hsu (National Taiwan University, Taiwan)
Daniel Kudenko (University of York, UK)
Paul Scerri (Carnegie Mellon University, USA)

Senior Program Committee

Sherief Abdallah (British University in Dubai)
Adrian Agogino (University of California, Santa Cruz)
Stéphane Airiau (University of Amsterdam)
Francesco Amigoni (Politecnico di Milano)
Ana Bazzan (Universidade Federal do Rio Grande do Sul)
Jamal Bentahar (Concordia University)
Rafael Bordini (Universidade Federal do Rio Grande do Sul)
Cristiano Castelfranchi (ISTC-CNR)
Steve Chien (Jet Propulsion Laboratory, Caltech)
Amit Chopra (University of Trento)
Brad Clement (California Institute of Technology)
Helder Coelho (Universidade de Lisboa)
Vincent Conitzer (Duke University)
Mehdi Dastani (Utrecht University)
Keith Decker (University of Delaware)
Ed Durfee (University of Michigan)
Edith Elkind (Nanyang Technological University)
Ulle Endriss (University of Amsterdam)
Piotr Gmytrasiewicz (University of Illinois at Chicago)
Jonathan Gratch (University of Southern California)
Dominic Greenwood (Whitestein Technologies)
Dirk Heylen (University of Twente)
Koen Hindriks (Delft University of Technology)
Takayuki Ito (Nagoya Institute of Technology)
Odest Jenkins (Brown University)
Gal Kaminka (Bar Ilan University)
Jeffrey Kephart (IBM Research)
Sven Koenig (University of Southern California)
Sarit Kraus (Bar-Ilan University)
Kate Larson (University of Waterloo)
João Leite (Universidade Nova de Lisboa)
Pedro Lima (Lisbon Technical University)
Michael Luck (King's College London)
Rajiv Maheswaran (University of Southern California)
Janusz Marecki (IBM Research)
Stacy Marsella (University of Southern California)
John-Jules Meyer (Utrecht University)

Daniele Nardi (Sapienza University Roma)
Ann Nowe (Vrije Universiteit Brussel)
Ana Paiva (INESC-ID)
Simon Parsons (City University of New York)
Michal Pechoucek (Czech Technical University)
Paul Piwek (The Open University)
Helmut Prendinger (National Institute of Informatics)
Iyad Rahwan (Masdar Institute)
Mark Riedl (Georgia Institute of Technology)
Thomas Rist (University of Applied Sciences Augsburg)
Juan Antonio Rodríguez-Aguilar (IIIA-CSIC)
Alex Rogers (University of Southampton)
Jeffrey Rosenschein (Hebrew University of Jerusalem)
Jordi Sabater-Mir (IIIA-CSIC)
Erol Şahin (Middle East Technical University)
Paul Scerri (Carnegie Mellon University)
Nathan Schurr (Aptima, Inc.)
Sandip Sen (University of Tulsa)
Murat Sensoy (University of Aberdeen)
Maarten Sierhuis (Palo Alto Research Center)
Carles Sierra (IIIA-CSIC)
Munindar Singh (North Carolina State University)
Elizabeth Sklar (City University of New York)
Katia Sycara (Carnegie Mellon University)
Matthew Taylor (Lafayette College)
John Thangarajah (Royal Melbourne Institute of Technology)
Simon Thompson (BT Research and Technology)
Paolo Torroni (University of Bologna)
Karl Tuyls (Maastricht University)
Wiebe van der Hoek (University of Liverpool)
M. Birna van Riemsdijk (Delft University of Technology)
Pradeep Varakantham (Singapore Management University)
Manuela Veloso (Carnegie Mellon University)
Katja Verbeeck (Katholieke Hogeschool Sint-Lieven)
Hannes Vilhjálmsson (Reykjavik University)
Michael Wellman (University of Michigan)
Steven Willmott (3scale Networks)
Michael Wooldridge (University of Liverpool)
Neil Yorke-Smith (American University of Beirut)
R. Michael Young (North Carolina State University)
Shlomo Zilberstein (University of Massachusetts Amherst)

Program Committee

Thomas Ågotnes (University of Bergen)
Noa Agmon (University of Texas, Austin)
H. Levent Akin (Bogaziçi University)
Marco Alberti (New University of Lisbon)
Huib Aldewereld (Utrecht University)
Natasha Alechina (University of Nottingham)
Martin Allen (University of Wisconsin-La Crosse)
Christopher Amato (Aptima Inc.)
Leila Amgoud (Institut de Recherche en Informatique de Toulouse)

Bo An (University of Massachusetts Amherst)
Giulia Andrighetto (ISTC-CNR)
Luis Antunes (Universidade de Lisboa)
Alexander Artikis (NCSR Demokritos)
Itai Ashlagi (Massachusetts Institute of Technology)
Katie Atkinson (University of Liverpool)
James Atlas (University of Delaware)
Ruth Aylett (Heriot-Watt University)
Yoram Bachrach (Microsoft Research)
Byung-Chull Bae (Samsung Advanced Institute of Technology)
Quan Bai (Tasmanian ICT Centre, CSIRO)
Matteo Baldoni (University di Torino)
João Balsa (Universidade de Lisboa)
Bikramjit Banerjee (University of Southern Mississippi)
Laura Barbulescu (Carnegie Mellon University)
Cristina Baroglio (University of Torino)
Tony Barrett (Jet Propulsion Laboratory)
Christian Becker-Asano (University of Freiburg)
Reinaldo Bianchi (FEI)
Mauro Birattari (Université Libre de Bruxelles)
Olivier Boissier (Ecole des Mines de Saint-Etienne)
Andrea Bonarini (Politecnico di Milano)
Tibor Bosse (Vrije Universiteit Amsterdam)
Luis Botelho (Instituto Universitário de Lisboa)
Sylvain Bouveret (ONERA)
Emma Bowring (University of the Pacific)
Ronen Brafman (Ben Gurion University)
Lars Braubach (University of Hamburg)
Joost Broekens (Delft University of Technology)
Brett Browning (Carnegie Mellon University)
Paul Buhler (College of Charleston)
Bernard Burg (Panasonic Laboratories)
Juan Burguillo (University of Vigo)
Birgit Burmeister (Daimler AG)
Lucian Busoniu (Delft University of Technology)
Zhongtang Cai (Oracle)
Daniele Calisi (Sapienza University of Rome)
Monique Calisti (Martel Consulting)
Tran Cao Son (New Mexico State University)
Javier Carbo (University Carlos III)
Alan Carlin (University of Massachusetts)
Stefano Carpin (University of California, Merced)
José Cascalho (Universidade dos Açores)
Marc Cavazza (University of Teesside)
Jesus Cerquides (IIIA-CSIC)
Brahim Chaib-draa (Laval University)
Georgios Chalkiadakis (University of Southampton)
Wei Chen (Intelligent Automation)
Xiaoping Chen (University of Science and Technology of China)
Shih-Fen Cheng (Singapore Management University)
Yun-Gyung Cheong (IT University of Copenhagen)
Sonia Chernova (Worcester Polytechnic Institute)
Carlos Iván Chesñevar (Universidad Nacional del Sur)
Federico Chesani (University of Bologna)
Maria Chli (Aston University)

Robin Cohen (University of Waterloo)
Nikolaus Correll (University of Colorado, Boulder)
Jacob Crandall (Masdar Institute)
Dominik Dahlem (Massachusetts Institute of Technology)
Esther David (Ashkelon College, Israel)
Célia da Costa Pereira (Université de Nice Sophia-Antipolis)
Antônio Carlos da Rocha Costa (Universidade Federal do Rio Grande)
Paulo Pinheiro da Silva (University of Texas, El Paso)
Scott DeLoach (Kansas State University)
Yves Demazeau (LIG-CNRS)
Louise Dennis (University of Liverpool)
Patrick De Causmaecker (Katholieke Universiteit Leuven)
Steven de Jong (Maastricht University)
Stefan De Wannemaecker (Katholieke Universiteit Leuven)
Mathijs de Weerd (Delft University of Technology)
Mary Bernardine Dias (Carnegie Mellon University)
Frank Dignum (Utrecht University)
Virginia Dignum (Delft University of Technology)
Oğuz Dikenelli (Ege University)
Jürgen Dix (Clausthal University of Technology)
Dmitri Dolgov (Google)
Klaus Dorer (Offenburg University)
Prashant Doshi (Univ of Georgia)
Paul Dunne (University of Liverpool)
Marc Esteva (IIIA-CSIC)
Piotr Faliszewski (AGH University of Science and Technology)
Alessandro Farinelli (University of Verona)
Shaheen Fatima (Loughborough University)
Sevan Ficici (Natural Selection)
Felix Fischer (Harvard SEAS)
Klaus Fischer (DFKI)
Nicoletta Fornara (Università della Svizzera Italiana)
Alex Fukunaga (University of Tokyo)
Naoki Fukuta (Shizuoka University)
Alberto Valero Gómez (University Carlos III de Madrid)
Thomas Gabel (Albert-Ludwigs-University Freiburg)
Kobi Gal (Ben-Gurion University of the Negev)
Nicola Gatti (Politecnico di Milano)
Patrick Gebhard (DFKI)
Enrico Gerding (University of Southampton)
Aditya K. Ghose (University of Wollongong)
Marco Gilles (Goldsmiths, University of London)
Paolo Giorgini (University of Trento)
Andrea Giovannucci (Princeton University)
Claudia Goldman (General Motors, Israel)
Valentin Goranko (Technical University of Denmark)
Guido Governatori (NICTA)
Adela Grando (University of Edinburgh)
Gianluigi Greco (University of Calabria)
Rachel Greenstadt (Drexel University)
Nathan Griffiths (University of Warwick)
Davide Grossi (University of Amsterdam)
Marek Grzes (University of Waterloo)
Mingyu Guo (University of Liverpool)
Christian Guttmann (EBTIC)

Jomi Hübner (Federal University of Santa Catarina)
James Hanson (IBM Research)
James Harland (Royal Melbourne Institute of Technology)
Paul Harrenstein (Technische Universität München)
Hiromitsu Hattori (Kyoto University)
Christopher Hazard (North Carolina State University)
Tarek Helmy (King Fahd University of Petroleum and Mineral)
Annerieke Heuvelink (TNO, Netherlands)
Sarah Hickmott (RMIT University)
Martin Hofmann (Lockheed Martin)
Mark Hoogendoorn (Vrije Universiteit)
Ian Horswill (Northwestern University)
Kaijen Hsiao (Willow Garage)
Michael Huhns (University of South Carolina)
Joris Hulstijn (Vrije Universiteit Amsterdam)
Luca Iocchi (Sapienza University Roma)
Michal Jakob (FEE Czech Technical University)
Wojciech Jamroga (University of Luxembourg)
Gaya Jayatilleke (Royal Melbourne Institute of Technology)
Arnav Jhala (University of California Santa Cruz)
Catholijn Jonker (Delft University of Technology)
Meir Kalech (Ben-Gurion University)
Marcelo Kallmann (University of California, Merced)
Ece Kamar (Microsoft Research)
Sachin Kamboj (University of Delaware)
Georgia Kastidou (University of Waterloo)
Takahiro Kawamura (Toshiba)
Michael Kipp (DFKI)
Alexandra Kirsch (Technische Universität München)
Franziska Klügl (Örebro University)
Alexander Kleiner (University of Freiburg)
Tomas Klos (Delft University of Technology)
Matthias Klusch (DFKI)
Matthew Knudson (Oregon State University)
Robert Kohout (DARPA)
Martin Kollingbaum (University of Aberdeen)
Sebastien Konieczny (CRIL-CNRS)
Stefan Kopp (Bielefeld University)
Gerhard Kraetzschmar (Bonn-Rhine-Sieg University of Applied Science)
Emiel Krahmer (Tilburg University)
Brigitte Krenn (Austrian Research Institute for Artificial Intelligence)
Daniel Kudenko (University of York)
Ugur Kuter (University of Maryland)
Miguel Ángel López Carmona (Universidad de Alcalá)
Michail Lagoudakis (Technical University of Crete)
Sebastien Lahaie (Yahoo! Research)
Luis Lamb (UFRGS)
Martin Lauer (Karlsruher Institut für Technologie)
Alessandro Lazaric (SequeL)
Samuel Leong (Microsoft Research)
Yves Lespérance (York University)
Victor Lesser (University of Massachusetts, Amherst)
Maxim Likachev (Carnegie Mellon University)
Wei Liu (University Western Australia)
Yaxin Liu (Google)

Brian Logan (University of Nottingham)
Alessio R. Lomuscio (Imperial College London)
Emiliano Lorini (Institut de Recherche en Informatique de Toulouse)
Bryan Kian Hsiang Low (National University of Singapore)
Zakaria Maamar (Zayed University)
Brian Magerko (Georgia Institute of Technology)
Roger Mailler (University of Tulsa)
Wenji Mao (Chinese Academy of Sciences)
Vangelis Markakis (Athens University of Economics and Business)
Lino Marques (University of Coimbra)
Viviana Mascardi (Universita' degli Studi di Genova)
Shigeo Matsubara (Kyoto University)
Tokuro Matsuo (Yamagata University)
Nicolas Maudet (University Paris-Dauphine)
Francisco Melo (INESC-ID/Instituto Superior Técnico)
Felipe Meneguzzi (Carnegie Mellon University)
Pedro Meseguer (IIIA CSIC)
Tomasz Michalak (University of Southampton)
Martin Michalowski (Adventium Labs)
Simon Miles (King's College London)
Dejan Milutinovic (University of California Santa Cruz)
Sanjay Modgil (King's College London)
Iqbal Mohamed (IBM Research)
Luis Moniz (Universidade de Lisboa)
Marco Montali (University of Bologna)
Bradford Mott (North Carolina State University)
Abdel-Ilhah Mouaddib (University of Caen Basse-Normandie)
Hideyuki Nakanishi (Osaka University)
Yukiko Nakano (Seikei University)
Nanjangud Narendra (IBM Research)
Toyoaki Nishida (Kyoto University)
Pablo Noriega (IIIA-CSIC)
Timothy Norman (University of Aberdeen)
Colm O'Riordan (National University of Ireland)
Magalie Ochs (CNRS)
Jean Oh (Carnegie Mellon University)
Frans Oliehoek (Massachusetts Institute of Technology)
Nir Oren (University of Aberdeen)
Mehmet Orgun (Macquarie University)
Charlie Ortiz (SRI International)
Sarah Osentoski (Brown University)
Sascha Ossowski (University Rey Juan Carlos)
Liviu Panait (Google)
Mario Paolucci (ISTC-CNR)
Dmitrii Pasechnik (Nanyang Technological University)
Terry Payne (University of Liverpool)
Catherine Pelachaud (Telecom ParisTech)
Johannes Pellenz (Federal Office of Defence Technology)
Marek Petrik (IBM Research)
Stacy Pfautz (Aptima)
Maria Silvia Pini (University of Padova)
Michael Pirker (Siemens)
Jeremy Pitt (Imperial College London)
Alexander Pokahr (University of Hamburg)
Daniel Polani (University of Hertfordshire)

Faruk Polat (Middle East Technical University)
Maria Polukarov (University of Southampton)
Enrico Pontelli (New Mexico State University)
Ronald Poppe (University of Twente)
Daniele Porello (University of Amsterdam)
Han La Poutré (Centrum Wiskunde and Informatica)
Rui Prada (INESC-ID and Instituto Superior Técnico)
Henry Prakken (Utrecht University)
Doina Precup (McGill University)
Ariel D. Procaccia (Harvard University)
Scott Proper (Oregon State University)
David Pynadath (University of Southern California)
Michael Quinlan (University of Texas, Austin)
Anita Raja (University of North Carolina at Charlotte)
Célia Ghedini Ralha (University of Brasília)
Sarvapali Ramchurn (University of Southampton)
Matthias Rehm (Aalborg University)
Dennis Reidsma (University of Twente)
Fenghui Ren (University of Wollongong)
Marcello Restelli (Politecnico di Milano)
Fernando Ribeiro (Universidade do Minho)
Alessandro Ricci (University of Bologna)
Debbie Richards (Macquarie University)
Giovanni Rimassa (Whitestein Technologies AG)
David Roberts (North Carolina State University)
Valentin Robu (University of Southampton)
Odinaldo Rodrigues (King's College London)
Nico Roos (Maastricht University)
Antonino Rotolo (University of Bologna)
Michael Rovatsos (University of Edinburgh)
Zachary Rubinstein (Carnegie Mellon University)
Wheeler Ruml (University of New Hampshire)
Vasile Rus (University of Memphis)
Zsófia Ruttkay (Moholy-Nagy University of Arts and Design Budapest)
Fariba Sadri (Imperial College London)
Alessandro Saffiotti (Orebro University)
Ken Satoh (National Institute of Informatics, Japan)
Martijn Schut (Vrije Universiteit)
Steven Shapiro (RMIT University)
Alexei Sharpanskykh (Vrije Universiteit Amsterdam)
Onn Shehory (IBM Research)
Dylan Shell (Texas A&M University)
Jiaying Shen (SRI International)
Mei Si (Rensselaer Polytechnic Institute)
Marius Silaghi (FIT)
Barry G. Silverman (University Pennsylvania)
Guillermo Simari (Universidad Nacional del Sur)
Stephen Smith (Carnegie Mellon University)
Leen-Kiat Soh (University of Nebraska-Lincoln)
Matthijs Spaan (Instituto Superior Técnico)
Mudhakar Srivatsa (IBM Research)
Jordan Srouf (American University of Beirut)
Eugen Staab (IMC)
Sebastian Stein (University of Southampton)
Gerald Steinbauer (Graz University of Technology)

Bas Steunebrink (Dalle Molle Institute for Artificial Intelligence)
Nathan Sturtevant (University of Denver)
Gita Sukthankar (University of Central Florida)
Evan Sultanik (Drexel University)
Pedro Szekely (USC Information Sciences Institute)
Juan Tapiador (University of York)
Luke Teacy (University of Ulster)
Andrea Tettamanzi (Università degli Studi di Milano)
Michael Thielscher (The University of New South Wales)
Andrea Thomaz (Georgia Institute of Technology)
Ingo Timm (University of Trier)
Viviane Torres da Silva (Universidade Federal Fluminense)
Jan Treur (Vrije Universiteit Amsterdam)
Paulo Trigo (Instituto Superior de Engenharia de Lisboa)
Nicolas Troquard (University of Liverpool)
Khiet Truong (University of Twente)
Takahiro Uchiya (Nagoya Institute of Technology)
Joel Uckelman (University of Amsterdam)
Paulo Urbano (FCUL)
Greet Vanden Berghe (Katholieke Hogeschool Sint-Lieven)
Kees van Deemter (University of Aberdeen)
Ielka van der Sluis (Trinity College Dublin)
Leendert van der Torre (University of Luxembourg)
Jurriaan van Diggelen (TNO, The Netherlands)
Betsy van Dijk (University of Twente)
Hans van Ditmarsch (University of Sevilla)
H. Van Dyke Parunak (Vector Research Center)
Rogier van Eijk (Utrecht University)
Wamberto Vasconcelos (University of Aberdeen)
Kristen Brent Venable (University of Padova)
Laurent Vercouter (Ecole des Mines de Saint-Etienne)
Jose Vidal (University of South Carolina)
Vinoba Vinayagamoorthy (BBC Research & Development)
Ubbo Visser (University of Miami)
Yevgeniy Vorobeychik (Sandia National Labs)
George Vouros (University of the Aegean)
Peter Vranx (Vrije Universiteit Brussel)
Yonghong Wang (Carnegie Mellon University)
Nick Webb (University at Albany)
Gerhard Weiss (University of Maastricht)
Danny Weyns (Katholieke Universiteit Leuven)
Michael Winikoff (University of Otago)
Cees Witteveen (Delft University of Technology)
Yang Xu (University Electronic Science and Tech of China)
Osher Yadgar (SRI International)
Hamdi Yahyaoui (Kuwait University)
Hirofumi Yamaki (Nagoya University)
Gaku Yamamoto (IBM Japan)
William Yeoh (University of Massachusetts)
Makoto Yokoo (Kyushu University)
Muhammad Younas (Oxford Brookes University)
Jie Zhang (Nanyang Technological University)
Minjie Zhang (The University of Wollongong)
Rong Zhou (PARC)
Ingrid Zukerman (Monash University)

Auxiliary Reviewers

John Augustine
Azizi ab Aziz
Haris Aziz
Aijun Bai
Tim Baarslag
Alexandros Belesiotis
Elizabeth Black
Thomas Bolander
Fiemke Both
Christoph Broschinski
Hendrik Buschmeier
Laurent Charlin
George Christelis
Evan Clark
Matt Crosby
Phan Minh Dung
Eliseo Ferrante
Francesco Figari
Jan-Gregor Fischer
Alexander Grushin
David Ben Hamo
Andreas Hertle
Greg Hines
Yazhou Huang
S. Waqar Jaffry
Thomas Keller
Eliahu Khalastchi
Yoonheui Kim
Ramachandra Kota
Rianne van Lambalgen

Steffen Lamparter
Viliam Lisý
Mentar Mahmudi
Robbert-Jan Merk
João Messias
Victor Naroditskiy
Christian Pietsch
Giovanni Pini
Matthijs Pontier
Evangelia Pyrga
Shulamit Reches
Inmaculada Rodriguez
Amir Sadeghipour
Hans Georg Seedig
Becher Silvio
Alexander Skopalik
Roni Stern
Ali Emre Turgut
Iris van de Kieft
Natalie van der Wal
Meritxell Vinyals
Wietske Visser
Grant Weddell
Matthew Whitaker
Simon Williamson
Feng Wu
Jiongkun Xie
Ramind Yaghubzadeh
Zongzhang Zhang

Sponsors

We would like to thank the following for their contribution to the success of this conference:

The International Foundation for Autonomous
Agents and Multiagent Systems



Platinum Sponsor

Artificial Intelligence Journal



Gold Sponsor

Agreement Technologies



Asian Office of Aerospace
Research & Development



Silver Sponsors

Etisalat British Telecom Innovation Centre



Foundation for Intelligent Physical Agents



IBM Research

IBM Research

Journal of Autonomous Agents and
Multi-Agent Systems



Wiley-Blackwell



Other Sponsors

IOS Press



Local Sponsors

National Science Council



Ministry of Education,
Republic of China (Taiwan)



Ministry of Foreign Affairs,
Republic of China (Taiwan)



National Tsing Hua University



Taiwanese Association of Artificial Intelligence



Contents

Main Program – Best Papers

Best Papers Session I

Agent-Based Control for Decentralised Demand Side Management in the Smart Grid <i>Sarvapali D. Ramchurn, Perukrishnen Vytelingum, Alex Rogers, Nicholas R. Jennings</i>	5
Deploying Power Grid-Integrated Electric Vehicles as a Multi-Agent System <i>Sachin Kamboj, Willett Kempton, Keith S. Decker</i>	13
Multi-Agent Monte Carlo Go <i>Leandro Soriano Marcolino, Hitoshi Matsubara</i>	21
Towards a Unifying Characterization for Quantifying Weak Coupling in Dec-POMDPs <i>Stefan J. Witwicki, Edmund H. Durfee</i>	29
GUARDS - Game Theoretic Security Allocation on a National Scale <i>James Pita, Milind Tambe, Christopher Kiekintveld, Shane Cullen, Erin Steigerwald</i>	37

Best Papers Session II

On the Outcomes of Multiparty Persuasion <i>Elise Bonzon, Nicolas Maudet</i>	47
Arbitrators in Overlapping Coalition Formation Games <i>Yair Zick, Edith Elkind</i>	55
Learning the Demand Curve in Posted-Price Digital Goods Auctions <i>Meenal Chhabra, Sanmay Das</i>	63
Ties Matter: Complexity of Voting Manipulation Revisited <i>Svetlana Obraztsova, Edith Elkind, Noam Hazon</i>	71
Designing Incentives for Boolean Games <i>Ulle Endriss, Sarit Kraus, Jérôme Lang, Michael Wooldridge</i>	79

Main Program – Full Papers

Session A1 – Robotics

Who Goes There? Selecting a Robot to Reach a Goal Using Social Regret <i>Meytal Traub, Gal A. Kaminka, Noa Agmon</i>	91
Exploration Strategies Based on Multi-Criteria Decision Making for Search and Rescue Autonomous Robots <i>Nicola Basilico, Francesco Amigoni</i>	99
Simulation-based Temporal Projection of Everyday Robot Object Manipulation <i>Lars Kunze, Mihai Emanuel Dolha, Emitza Guzman, Michael Beetz</i>	107
Online Anomaly Detection in Unmanned Vehicles <i>Eliahu Khalastchi, Gal A. Kaminka, Meir Kalech, Raz Lin</i>	115
Tree Adaptive A* <i>Carlos Hernández, Xiaoxun Sun, Sven Koenig, Pedro Meseguer</i>	123

Session B1 – Distributed Problem Solving I

Quality Guarantees for Region Optimal DCOP Algorithms <i>Meritxell Vinyals, Eric Shieh, Jesus Cerquides, Juan Antonio Rodriguez-Aguilar, Zhengyu Yin, Milind Tambe, Emma Bowring</i>	133
Distributed Algorithms for Solving the Multiagent Temporal Decoupling Problem <i>James C. Boerkoel, Edmund H. Durfee</i>	141

Decomposing Constraint Systems: Equivalences and Computational Properties <i>Wiebe van der Hoek, Cees Witteveen, Michael Wooldridge</i>	149
Decentralized Monitoring of Distributed Anytime Algorithms <i>Alan Carlin, Shlomo Zilberstein</i>	157
Consensus Acceleration in Multiagent Systems with the Chebyshev Semi-Iterative Method <i>Renato L.G. Cavalcante, Alex Rogers, Nicholas R. Jennings</i>	165
Session C1 – Game Theory I	
Information Elicitation for Decision Making <i>Yiling Chen, Ian A. Kash</i>	175
Stable Partitions in Additively Separable Hedonic Games <i>Haris Aziz, Felix Brandt, Hans Georg Seedig</i>	183
Complexity of Coalition Structure Generation <i>Haris Aziz, Bart de Keijzer</i>	191
Equilibrium Approximation in Simulation-Based Extensive-Form Games <i>Nicola Gatti, Marcello Restelli</i>	199
Maximum Causal Entropy Correlated Equilibria for Markov Games <i>Brian D. Ziebart, J. Andrew Bagnell, Anind K. Dey</i>	207
Session D1 – Multiagent Learning	
Learning Action Models for Multi-Agent Planning <i>Hankz Hankui Zhuo, Hector Muñoz-Avila, Qiang Yang</i>	217
Theoretical Considerations of Potential-Based Reward Shaping for Multi-Agent Systems <i>Sam Devlin, Daniel Kudenko</i>	225
Evolving Subjective Utilities: Prisoner’s Dilemma Game Examples <i>Koichi Moriyama, Satoshi Kurihara, Masayuki Numao</i>	233
Cooperation through Reciprocity in Multiagent Systems: An Evolutionary Analysis <i>Christian Hütter, Klemens Böhm</i>	241
Distributed Cooperation in Wireless Sensor Networks <i>Mihail Mihaylov, Yann-Aël Le Borgne, Karl Tuyls, Ann Nowé</i>	249
Session A2 – Logic-Based Approaches I	
A Framework for Coalitional Normative Systems <i>Jun Wu, Chongjun Wang, Junyuan Xie</i>	259
Practical Argumentation Semantics for Socially Efficient Defeasible Consequence <i>Hiroyuki Kido, Katsumi Nitta</i>	267
Taming the Complexity of Linear Time BDI Logics <i>Nils Bulling, Koen V. Hindriks</i>	275
Session B2 – Agent-Based System Development I	
Scenarios for System Requirements Traceability and Testing <i>John Thangarajah, Gaya Jayatilleke, Lin Padgham</i>	285
Kokomo: An Empirically Evaluated Methodology for Affective Applications <i>Derek J. Sollenberger, Munindar P. Singh</i>	293
Programming Mental State Abduction <i>Michal Sindlar, Mehdi Dastani, John-Jules Ch. Meyer</i>	301
Session C2 – Social Choice Theory	
Possible And Necessary Winners In Voting Trees: Majority Graphs Vs. Profiles <i>Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, Toby Walsh</i>	311
Tight Bounds for Strategyproof Classification <i>Reshef Meir, Shaull Almagor, Assaf Michaely, Jeffrey S. Rosenschein</i>	319

A Double Oracle Algorithm for Zero-Sum Security Games on Graphs <i>Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, Milind Tambe</i>	327
Session D2 – Preferences and Strategies	
Modeling Social Preferences in Multi-player Games <i>Brandon Wilson, Inon Zuckerman, Dana Nau</i>	337
A Study of Computational and Human Strategies in Revelation Games <i>Noam Peled, Ya'akov (Kobi) Gal, Sarit Kraus</i>	345
Efficient Heuristic Approach to Dominance Testing in CP-nets <i>Minyi Li, Quoc Bao Vo, Ryszard Kowalczyk</i>	353
Session A3 – Distributed Problem Solving II	
Resource-Aware Junction Trees for Efficient Multi-Agent Coordination <i>N. Stefanovitch, A. Farinelli, Alex Rogers, Nicholas R. Jennings</i>	363
Bounded Decentralised Coordination over Multiple Objectives <i>Francesco M. Delle Fave, Ruben Stranders, Alex Rogers, Nicholas R. Jennings</i>	371
Communication-Constrained DCOPs: Message Approximation in GDL with Function Filtering <i>Marc Pujol-Gonzalez, Jesus Cerquides, Pedro Meseguer, Juan Antonio Rodriguez-Aguilar</i>	379
Session B3 – Agent-Based System Development II	
AgentScope: Multi-Agent Systems Development in Focus <i>Elth Ogston, Frances Brazier</i>	389
Agent Programming with Priorities and Deadlines <i>Konstantin Vikhorev, Natasha Alechina, Brian Logan</i>	397
Rich Goal Types in Agent Programming <i>Mehdi Dastani, M. Birna van Riemsdijk, Michael Winikoff</i>	405
Session C3 – Bounded Rationality	
Expert-Mediated Search <i>Meenal Chhabra, Sanmay Das, David Sarne</i>	415
Using Aspiration Adaptation Theory to Improve Learning <i>Avi Rosenfeld, Sarit Kraus</i>	423
Less Is More: Restructuring Decisions to Improve Agent Search <i>David Sarne, Avshalom Elmalech, Barbara J. Grosz, Moti Geva</i>	431
Session D3 – Virtual Agents I	
Culture-related Differences in Aspects of Behavior for Virtual Characters Across Germany and Japan <i>Birgit Endrass, Elisabeth André, Afia Akhter Lipi, Matthias Rehm, Yukiko Nakano</i>	441
Controlling Narrative Time in Interactive Storytelling <i>Julie Porteous, Jonathan Teutenberg, Fred Charles, Marc Cavazza</i>	449
ESCAPES - Evacuation Simulation with Children, Authorities, Parents, Emotions, and Social comparison <i>Jason Tsai, Natalie Fridman, Emma Bowring, Matthew Brown, Shira Epstein, Gal A. Kaminka, Stacy Marsella, Andrew Ogden, Inbal Rika, Ankur Sheel, Matthew E. Taylor, Xuezhi Wang, Avishay Zilka, Milind Tambe</i>	457
Session A4 – Agent Communication	
Commitments with Regulations: Reasoning about Safety and Control in REGULA <i>Elisa Marengo, Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, Viviana Patti, Munindar P. Singh</i>	467
Specifying and Applying Commitment-Based Business Patterns <i>Amit K. Chopra, Munindar P. Singh</i>	475

On the Verification of Social Commitments and Time <i>Mohamed El-Menshawy, Jamal Bentahar, Hongyang Qu, Rachida Dssouli</i>	483
Information-Driven Interaction-Oriented Programming: BSPL, the Blindingly Simple Protocol Language <i>Munindar P. Singh</i>	491
On Topic Selection Strategies in Multi-Agent Naming Game <i>Wojciech Lorkiewicz, Ryszard Kowalczyk, Radoslaw Katarzyniak, Quoc Bao Vo</i>	499
Session B4 – Game Theory and Learning	
Reaching Correlated Equilibria Through Multi-agent Learning <i>Ludek Cigler, Boi Faltings</i>	509
Sequential Targeted Optimality as a New Criterion for Teaching and Following in Repeated Games <i>Max Knobbout, Gerard A.W. Vreeswijk</i>	517
On the Quality and Complexity of Pareto Equilibria in the Job Scheduling Game <i>Leah Epstein, Elena Kleiman</i>	525
Game Theory-Based Opponent Modeling in Large Imperfect-Information Games <i>Sam Ganzfried, Tuomas Sandholm</i>	533
False-name Bidding in First-price Combinatorial Auctions with Incomplete Information <i>Atsushi Iwasaki, Atsushi Katsuragi, Makoto Yokoo</i>	541
Session C4 – Teamwork	
Metastrategies in the Colored Trails Game <i>Steven de Jong, Daniel Hennes, Karl Tuyls, Ya’akov (Kobi) Gal</i>	551
Computing Stable Outcomes in Hedonic Games with Voting-Based Deviations <i>Martin Gairing, Rahul Savani</i>	559
Empirical Evaluation of Ad Hoc Teamwork in the Pursuit Domain <i>Samuel Barrett, Peter Stone, Sarit Kraus</i>	567
Decision Theoretic Behavior Composition <i>Nitin Yadav, Sebastian Sardina</i>	575
Solving Election Manipulation Using Integer Partitioning Problems <i>Andrew Lin</i>	583
Session A5 – Learning Agents	
Using Iterated Reasoning to Predict Opponent Strategies <i>Michael Wunder, John Robert Yaros, Michael Littman, Michael Kaisers</i>	593
Cognitive Policy Learner: Biasing Winning or Losing Strategies <i>Dominik Dahlem, Jim Dowling, William Harrison</i>	601
Agent-Mediated Multi-Step Optimization for Resource Allocation in Distributed Sensor Networks <i>Bo An, Victor Lesser, David Westbrook, Michael Zink</i>	609
Integrating Reinforcement Learning with Human Demonstrations of Varying Ability <i>Matthew E. Taylor, Halit Bener Suay, Sonia Chernova</i>	617
Session B5 – Auction and Incentive Design	
Incentive Design for Adaptive Agents <i>Yiling Chen, Jerry Kung, David C. Parkes, Ariel D. Procaccia, Haoqi Zhang</i>	627
A Truth Serum for Sharing Rewards <i>Arthur Carvalho, Kate Larson</i>	635
Capability-Aligned Matching: Improving Quality of Games with a Purpose <i>Che-Liang Chiou, Jane Yung-Jen Hsu</i>	643
False-name-proof Mechanism Design without Money <i>Taiki Todo, Atsushi Iwasaki, Makoto Yokoo</i>	651

Majority-Rule-Based Preference Aggregation on Multi-Attribute Domains with CP-Nets <i>Minyi Li, Quoc Bao Vo, Ryszard Kowalczyk</i>	659
Session C5 – Simulation and Emergence	
Emerging Cooperation on Complex Networks <i>Norman Salazar, Juan Antonio Rodriguez-Aguilar, Josep Lluís Arcos, Ana Peleteiro, Juan C. Burquillo-Rial</i>	669
An Investigation of the Vulnerabilities of Scale Invariant Dynamics in Large Teams <i>Robin Grinton, Paul Scerri, Katia Sycara</i>	677
The Evolution of Cooperation in Self-Interested Agent Societies: A Critical Study <i>Lisa-Maria Hofmann, Nilanjan Chakraborty, Katia Sycara</i>	685
A Model of Norm Emergence and Innovation in Language Change <i>Samarth Swarup, Andrea Apolloni, Zsuzsanna Fagyal</i>	693
Dynamic Level of Detail for Large Scale Agent-Based Urban Simulations <i>Laurent Navarro, Fabien Flacher, Vincent Corruble</i>	701
Session D5 – Logic-Based Approaches II	
Reasoning About Local Properties in Modal Logic <i>Hans van Ditmarsch, Wiebe van der Hoek, Barteld Kooi</i>	711
Knowledge and Control <i>Wiebe van der Hoek, Nicolas Troquard, Michael Wooldridge</i>	719
Strategic Games and Truly Playable Effectivity Functions <i>Valentin Goranko, Wojciech Jamroga, Paolo Turrini</i>	727
Scientia Potentia Est <i>Thomas Ágotnes, Wiebe van der Hoek, Michael Wooldridge</i>	735
Tractable Model Checking for Fragments of Higher-Order Coalition Logic <i>Patrick Doherty, Barbara Dunin-Kępicz, Andrzej Szalas</i>	743
Session A6 – Robotics and Learning	
Active Markov Information-Theoretic Path Planning for Robotic Environmental Sensing <i>Kian Hsiang Low, John M. Dolan, Pradeep Khosla</i>	753
Horde: A Scalable Real-time Architecture for Learning Knowledge from Unsupervised Sensorimotor Interaction <i>Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, Doina Precup</i>	761
On Optimizing Interdependent Skills: A Case Study in Simulated 3D Humanoid Robot Soccer <i>Daniel Urieli, Patrick MacAlpine, Shivaram Kalyanakrishnan, Yimon Bentor, Peter Stone</i>	769
Metric Learning for Reinforcement Learning Agents <i>Matthew E. Taylor, Brian Kulis, Fei Sha</i>	777
Session B6 – Energy Applications	
Cooperatives of Distributed Energy Resources for Efficient Virtual Power Plants <i>Georgios Chalkiadakis, Valentin Robu, Ramachandra Kota, Alex Rogers, Nicholas R. Jennings</i>	787
How Agents Can Help Curbing Fuel Combustion – a Performance Study of Intersection Control for Fuel-Operated Vehicles <i>Natalja Pulter, Heiko Schepperle, Klemens Böhm</i>	795
Decentralized Coordination Of Plug-in Hybrid Vehicles For Imbalance Reduction In A Smart Grid <i>Stijn Vandael, Klaas De Craemer, Nelis Boucké, Tom Holwoet, Geert Deconinck</i>	803
Online Mechanism Design for Electric Vehicle Charging <i>Enrico H. Gerding, Valentin Robu, Sebastian Stein, David C. Parkes, Alex Rogers, Nicholas R. Jennings</i>	811

Session C6 – Voting Protocols

Homogeneity and Monotonicity of Distance-Rationalizable Voting Rules <i>Edith Elkind, Piotr Faliszewski, Arkadii Slinko</i>	821
Possible Winners When New Alternatives Join: New Results Coming Up! <i>Lirong Xia, Jérôme Lang, Jérôme Monnot</i>	829
The Complexity of Voter Partition in Bucklin and Fallback Voting: Solving Three Open Problems <i>Gábor Erdélyi, Lena Piras, Jörg Rothe</i>	837
An Algorithm for the Coalitional Manipulation Problem under Maximin <i>Michael Zuckerman, Omer Lev, Jeffrey S. Rosenschein</i>	845
Computational Complexity of Two Variants of the Possible Winner Problem <i>Dorothea Baumeister, Magnus Roos, Jörg Rothe</i>	853

Session D6 – Trust and Organisational Structure

Trust as Dependence: A Logical Approach <i>Munindar P. Singh</i>	863
Multi-Layer Cognitive Filtering by Behavioral Modeling <i>Zeinab Noorian, Stephen Marsh, Michael Fleming</i>	871
Argumentation-Based Reasoning in Agents with Varying Degrees of Trust <i>Simon Parsons, Yuqing Tang, Elizabeth Sklar, Peter McBurney, Kai Cai</i>	879
A Particle Filter for Bid Estimation in Ad Auctions with Periodic Ranking Observations <i>David Pardoe, Peter Stone</i>	887
Conviviality Measures <i>Patrice Caire, Baptiste Alcalde, Leendert van der Torre, Chattrakul Sombatheera</i>	895

Session A7 – Argumentation and Negotiation

Choosing Persuasive Arguments for Action <i>Elizabeth Black, Katie Atkinson</i>	905
Argumentation Strategies for Plan Resourcing <i>Chukwuemeka D. Emele, Timothy J. Norman, Simon Parsons</i>	913
Multi-Criteria Argument Selection In Persuasion Dialogues <i>Tom L. van der Weide, Frank Dignum, John-Jules Ch. Meyer, H. Prakken, Gerard A.W. Vreeswijk</i>	921
Analyzing Intra-Team Strategies for Agent-Based Negotiation Teams <i>Víctor Sánchez-Anguix, Vicente Julián, Vicente Botti, Ana García-Fornes</i>	929
The Effect of Expression of Anger and Happiness in Computer Agents on Negotiations with Humans <i>Celso M. de Melo, Peter Carnevale, Jonathan Gratch</i>	937

Session B7 – Planning

Toward Error-Bounded Algorithms for Infinite-Horizon DEC-POMDPs <i>Jilles S. Dibangoye, Abdel-Allah Mouaddib, Brahim Chaib-draa</i>	947
Distributed Model Shaping for Scaling to Decentralized POMDPs with Hundreds of Agents <i>Prasanna Velagapudi, Pradeep Varakantham, Katia Sycara, Paul Scerri</i>	955
Efficient Planning in R-max <i>Marek Grzes, Jesse Hoey</i>	963
Multiagent Argumentation for Cooperative Planning in DeLP-POP <i>Pere Pardo, Sergio Pajares, Eva Onaindia, Pilar Dellunde, Lluís Godo</i>	971

Session C7 – Game Theory II

Computing a Self-Confirming Equilibrium in Two-Player Extensive-Form Games <i>Nicola Gatti, Fabio Panozzo, Sofia Ceppi</i>	981
Computing Time-Dependent Policies for Patrolling Games with Mobile Targets <i>Branislav Bošanský, Viliam Lisý, Michal Jakob, Michal Pěchouček</i>	989

Quality-bounded Solutions for Finite Bayesian Stackelberg Games: Scaling up <i>Manish Jain, Christopher Kiekintveld, Milind Tambe</i>	997
Approximation Methods for Infinite Bayesian Stackelberg Games: Modeling Distributional Payoff Uncertainty <i>Christopher Kiekintveld, Janusz Marecki, Milind Tambe</i>	1005
Solving Stackelberg Games with Uncertain Observability <i>Dmytro Korzhyk, Vincent Conitzer, Ronald Parr</i>	1013

Session D7 – Virtual Agents II

A Style Controller for Generating Virtual Human Behaviors <i>Chung-Cheng Chiu, Stacy Marsella</i>	1023
The Face of Emotions: A Logical Formalization of Expressive Speech Acts <i>Nadine Guiraud, Dominique Longin, Emiliano Lorini, Sylvie Pesty, Jérémy Rivière</i>	1031
I’ve Been Here Before! Location and Appraisal in Memory Retrieval <i>Paulo F. Gomes, Carlos Martinho, Ana Paiva</i>	1039
From Body Space to Interaction Space - Modeling Spatial Cooperation for Virtual Humans <i>Nhung Nguyen, Ipke Wachsmuth</i>	1047
Effect of Time Delays on Agents’ Interaction Dynamics <i>Ken Prepin, Catherine Pelachaud</i>	1055

Main Program – Extended Abstracts

Red Session

A Computational Model of Achievement Motivation for Artificial Agents <i>Kathryn E. Merrick</i>	1067
Incremental DCOP Search Algorithms for Solving Dynamic DCOPs <i>William Yeoh, Pradeep Varakantham, Xiaoxun Sun, Sven Koenig</i>	1069
MetaTrust: Discriminant Analysis of Local Information for Global Trust Assessment <i>Liu Xin, Gilles Tredan, Anwitaman Datta</i>	1071
Efficient Penalty Scoring Functions for Group Decision-making with TCP-nets <i>Minyi Li, Quoc Bao Vo, Ryszard Kowalczyk</i>	1073
A Curious Agent for Network Anomaly Detection <i>Kamran Shafi, Kathryn E. Merrick</i>	1075
Agents, Pheromones, and Mean-Field Models <i>H. Van Dyke Parunak</i>	1077
Basis Function Discovery using Spectral Clustering and Bisimulation Metrics <i>Gheorghe Comanici, Doina Precup</i>	1079
Incentive Compatible Influence Maximization in Social Networks and Application to Viral Marketing <i>Mayur Mohite, Y. Narahari</i>	1081
On Optimal Agendas for Package Deal Negotiation <i>Shaheen Fatima, Michael Wooldridge, Nicholas R. Jennings</i>	1083
An Abstract Framework for Reasoning About Trust <i>Elisabetta Erriquez, Wiebe van der Hoek, Michael Wooldridge</i>	1085
Message-Passing Algorithms for Large Structured Decentralized POMDPs <i>Akshat Kumar, Shlomo Zilberstein</i>	1087
Jogger: Models for Context-Sensitive Reminding <i>Ece Kamar, Eric Horvitz</i>	1089
Spatio-Temporal A* Algorithms for Offline Multiple Mobile Robot Path Planning <i>Wenjie Wang, Wooi Boon Goh</i>	1091

Influence of Head Orientation in Perception of Personality Traits in Virtual Agents <i>Diana Arellano, Nikolaus Bee, Kathrin Janowski, Elisabeth André, Javier Varona, Francisco J. Perales</i>	1093
Conflict Resolution with Argumentation Dialogues <i>Xiuyi Fan, Francesca Toni</i>	1095
Reasoning Patterns in Bayesian Games <i>Dimitrios Antos, Avi Pfeffer</i>	1097
Using Coalitions of Wind Generators and Electric Vehicles for Effective Energy Market Participation <i>Matteo Vasirani, Ramachandra Kota, Renato L.G. Cavalcante, Sascha Ossowski, Nicholas R. Jennings</i>	1099
Negotiation Over Decommitment Penalty <i>Bo An, Victor Lesser</i>	1101
Ship Patrol: Multiagent Patrol under Complex Environmental Conditions <i>Noa Agmon, Daniel Urieli, Peter Stone</i>	1103
Empirical and Theoretical Support for Lenient Learning <i>Daan Bloembergen, Michael Kaisers, Karl Tuyls</i>	1105
A Formal Framework for Reasoning about Goal Interactions <i>Michael Winikoff</i>	1107
On-line Reasoning for Institutionally-Situated BDI agents <i>Tina Balke, Marina De Vos, Julian Padget, Dimitris Traskas</i>	1109
Strategy Purification <i>Sam Ganzfried, Tuomas Sandholm, Kevin Waugh</i>	1111
Agent-Based Container Terminal Optimisation <i>Stephen Cranefield, Roger Jarquin, Guannan Li, Brent Martin, Rainer Unland, Hanno-Felix Wagner, Michael Winikoff, Thomas Young</i>	1113
Solving Delayed Coordination Problems in MAS <i>Yann-Michaël De Hawwere, Peter Vrancx, Ann Nowé</i>	1115
Human-like Memory Retrieval Mechanisms for Social Companions <i>Mei Yü Lim, Ruth Aylett, Patricia A. Vargas, Wan Ching Ho, João Dias</i>	1117
Forgetting Through Generalisation - A Companion with Selective Memory <i>Mei Yü Lim, Ruth Aylett, Patricia A. Vargas, Sibylle Enz, Wan Ching Ho</i>	1119
Representation of Coalitional Games with Algebraic Decision Diagrams <i>Karthik .V. Aadithya, Tomasz P. Michalak, Nicholas R. Jennings</i>	1121
Game Theoretical Adaptation Model for Intrusion Detection System <i>Martin Rehak, Michal Pěchouček, Martin Grill, Jan Stiborek, Karel Bartos</i>	1123
Solving Strategic Bargaining with Arbitrary One-Sided Uncertainty <i>Sofia Ceppi, Nicola Gatti, Claudio Iuliano</i>	1125
Manipulation in Group Argument Evaluation <i>Martin Caminada, Gabriella Pigozzi, Mikolaj Podlaszewski</i>	1127
Abstraction for Model Checking Modular Interpreted Systems over ATL <i>Michael Köster, Peter Lohmann</i>	1129
VIXEE an Innovative Communication Infrastructure for Virtual Institutions <i>Tomas Trescak, Marc Esteva, Inmaculada Rodriguez</i>	1131
Smart Walkers! Enhancing the Mobility of the Elderly <i>Mathieu Sinn, Pascal Poupart</i>	1133
Modeling Empathy for a Virtual Human: How, When and to What Extent? <i>Hana Boukricha, Ipke Wachsmuth</i>	1135
Multi-Agent Abductive Reasoning with Confidentiality <i>Jiefei Ma, Alessandra Russo, Krysia Broda, Emil Lupu</i>	1137

Reasoning About Preferences in BDI Agent Systems <i>Simeon Visser, John Thangarajah, James Harland</i>	1139
---	------

Blue Session

Probabilistic Hierarchical Planning over MDPs <i>Yuqing Tang, Felipe Meneguzzi, Katia Sycara, Simon Parsons</i>	1143
Can Trust Increase the Efficiency of Cake Cutting Algorithms? <i>Roie Zivan</i>	1145
Decentralized Decision Support for an Agent Population in Dynamic and Uncertain Domains <i>Pradeep Varakantham, Shih-Fen Cheng, Nguyen Thi Duong</i>	1147
Adaptive Decision Support for Structured Organizations: A Case for OrgPOMDPs <i>Pradeep Varakantham, Nathan Schurr, Alan Carlin, Christopher Amato</i>	1149
iCLUB: An Integrated Clustering-Based Approach to Improve the Robustness of Reputation Systems <i>Siyuan Liu, Jie Zhang, Chunyan Miao, Yin-Leng Theng, Alex C. Kot</i>	1151
Effective Variants of Max-Sum Algorithm to Radar Coordination and Scheduling <i>Yoonheui Kim, Michael Krainin, Victor Lesser</i>	1153
Improved Computational Models of Human Behavior in Security Games <i>Rong Yang, Christopher Kiekintveld, Fernando Ordonez, Milind Tambe, Richard John</i>	1155
Agent-Based Resource Allocation in Dynamically Formed CubeSat Constellations <i>Chris HolmesParker, Adrian Agogino</i>	1157
A Simple Curious Agent to Help People be Curious <i>Han Yu, Zhiqi Shen, Chunyan Miao, Ah-Hwee Tan</i>	1159
Social Instruments for Convention Emergence <i>Daniel Villatoro, Jordi Sabater-Mir, Sandip Sen</i>	1161
Learning By Demonstration in Repeated Stochastic Games <i>Jacob W. Crandall, Malek H. Altakrori, Yomna M. Hassan</i>	1163
Maximizing Revenue in Symmetric Resource Allocation Systems When User Utilities Exhibit Diminishing Returns <i>Roie Zivan, Miroslav Dudík, Praveen Paruchuri, Katia Sycara</i>	1165
Collaborative Diagnosis of Exceptions to Contracts <i>Özgür Kafalı, Francesca Toni, Paolo Torroni</i>	1167
Genetic Algorithm Aided Optimization of Hierarchical Multiagent System Organization <i>Ling Yu, Zhiqi Shen, Chunyan Miao, Victor Lesser</i>	1169
Complexity of Multiagent BDI Logics with Restricted Modal Context <i>Marcin Dziubiński</i>	1171
Extension of MC-net-based Coalition Structure Generation: Handling Negative Rules and Externalities <i>Ryo Ichimura, Takato Hasegawa, Suguru Ueda, Atsushi Iwasaki, Makoto Yokoo</i>	1173
Diagnosing Commitments: Delegation Revisited <i>Özgür Kafalı, Paolo Torroni</i>	1175
ADAPT: Abstraction Hierarchies to Succinctly Model Teamwork <i>Meirav Hadad, Avi Rosenfeld</i>	1177
Rip-off: Playing the Cooperative Negotiation Game <i>Yoram Bachrach, Pushmeet Kohli, Thore Graepel</i>	1179
Interfacing a Cognitive Agent Platform with a Virtual World: a Case Study using Second Life <i>Surangika Ranathunga, Stephen Cranefield, Martin Purvis</i>	1181
Message-Generated Kripke Semantics <i>Jan van Eijck, Floor Sietsma</i>	1183
Substantiating Quality Goals with Field Data for Socially-Oriented Requirements Engineering <i>Sonja Pedell, Tim Müller, Leon Sterling, Frank Vetere, Steve Howard, Jeni Paay</i>	1185

Normative Programs and Normative Mechanism Design	
<i>Nils Bulling, Mehdi Dastani</i>	1187
Privacy-Intimacy Tradeoff in Self-disclosure	
<i>Jose M. Such, Agustin Espinosa, Ana García-Fornes, Carles Sierra</i>	1189
Reasoning About Norm Compliance	
<i>Natalia Criado, Estefania Argente, Vicente Botti, Pablo Noriega</i>	1191
Emergence of Norms for Social Efficiency in Partially Iterative Non-Coordinated Games	
<i>Toshiharu Sugawara</i>	1193
On the Construction of Joint Plans through Argumentation Schemes	
<i>Oscar Sapena, Alejandro Torreño, Eva Onaindia</i>	1195
Team Coverage Games	
<i>Yoram Bachrach, Pushmeet Kohli, Vladimir Kolmogorov</i>	1197
Agent-based Inter-Company Transport Optimization	
<i>Klaus Dorer, Ingo Schindler, Dominic Greenwood</i>	1199
Belief/Goal Sharing BDI Modules	
<i>Michal Cap, Mehdi Dastani, Maaïke Harbers</i>	1201
Neural Symbolic Architecture for Normative Agents	
<i>Guido Boella, Silvano Colombo Tosatto, Artur d'Avila Garcez, Valerio Genovese, Dino Ienco, Leendert van der Torre</i>	1203
No Smoking Here: Compliance Differences Between Legal and Social Norms	
<i>Francien Dechesne, Virginia Dignum</i>	1205
Agents That Speak: Modelling Communicative Plans and Information Sources in a Logic of Announcements	
<i>Philippe Balbiani, Nadine Guiraud, Andreas Herzig, Emiliano Lorini</i>	1207
Procedural Fairness in Stable Marriage Problems	
<i>Mirco Gelain, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, Toby Walsh</i>	1209
Tag-Based Cooperation in N-Player Dilemmas	
<i>Enda Howley, Jim Duggan</i>	1211
Heuristic Multiagent Planning with Self-Interested Agents	
<i>Matt Crosby, Michael Rovatsos</i>	1213
Mining Qualitative Context Models from Multiagent Interactions	
<i>Emilio Serrano, Michael Rovatsos, Juan Botia</i>	1215
Partially Observable Stochastic Game-based Multi-Agent Prediction Markets	
<i>Janyl Jumadinova, Prithviraj Dasgupta</i>	1217
Green Session	
A Cost-Based Transition Approach for Multiagent Systems Reorganization	
<i>Juan M. Alberola, Vicente Julián, Ana García-Fornes</i>	1221
Towards an Agent-Based Proxemic Model for Pedestrian and Group Dynamics: Motivations and First Experiments	
<i>Sara Manzoni, Giuseppe Vizzari, Kazumichi Ohtsuka, Kenichiro Shimura</i>	1223
Batch Reservations in Autonomous Intersection Management	
<i>Neda Shahidi, Tsz-Chiu Au, Peter Stone</i>	1225
Multi-Agent, Reward Shaping for RoboCup KeepAway	
<i>Sam Devlin, Marek Grześ, Daniel Kudenko</i>	1227
Approximating Behavioral Equivalence of Models Using Top-K Policy Paths	
<i>Yifeng Zeng, Yingke Chen, Prashant Doshi</i>	1229
Reflection about Capabilities for Role Enactment	
<i>M. Birna van Riemsdijk, Virginia Dignum, Catholijn M. Jonker, Huib Aldewereld</i>	1231

Prognostic Normative Reasoning in Coalition Planning	
<i>Jean Oh, Felipe Meneguzzi, Katia Sycara, Timothy J. Norman</i>	1233
Virtual Agent Perception in Large Scale Multi-Agent Based Simulation Systems	
<i>Dane Kuiper, Rym Z. Wenkstern</i>	1235
A Formal Analysis of the Outcomes of Argumentation-based Negotiations	
<i>Leila Amgoud, Srdjan Vesic</i>	1237
Modeling the Emergence of Norms	
<i>Logan Brooks, Wayne Iba, Sandip Sen</i>	1239
Introducing Homophily to Improve Semantic Service Search in a Self-adaptive System	
<i>E. del Val, M. Rebollo, Vicente Botti</i>	1241
Adaptive Regulation of Open MAS: an Incentive Mechanism based on Modifications of the Environment	
<i>Roberto Centeno, Holger Billhardt</i>	1243
Allocating Spatially Distributed Tasks in Large, Dynamic Robot Teams	
<i>Steven Okamoto, Nathan Brooks, Sean Owens, Katia Sycara, Paul Scerri</i>	1245
Bounded Optimal Team Coordination with Temporal Constraints and Delay Penalties	
<i>G. Ayorkor Korsah, Anthony Stentz, M. Bernardine Dias</i>	1247
A Perception Framework for Intelligent Characters in Serious Games	
<i>Joost van Oijen, Frank Dignum</i>	1249
SR-APL: A Model for a Programming Language for Rational BDI Agents with Prioritized Goals	
<i>Shakil M. Khan, Yves Lespérance</i>	1251
Designing Petri Net Supervisors for Multi-Agent Systems from LTL Specifications	
<i>Bruno Lacerda, Pedro U. Lima</i>	1253
Friend or Foe? Detecting an Opponent's Attitude in Normal Form Games	
<i>Steven Damer, Maria Gini</i>	1255
The BDI Driver in a Service City	
<i>Marco Lützenberger, Nils Masuch, Benjamin Hirsch, Sebastian Ahrndt, Axel Heßler, Sahin Albayrak</i>	1257
Identifying and Exploiting Weak-Information Inducing Actions in Solving POMDPs	
<i>Ekhlhas Sonu, Prashant Doshi</i>	1259
Teamwork in Distributed POMDPs: Execution-time Coordination Under Model Uncertainty	
<i>Jun-Young Kwak, Rong Yang, Zhengyu Yin, Matthew E. Taylor, Milind Tambe</i>	1261
Escaping Local Optima in POMDP Planning as Inference	
<i>Pascal Poupart, Tobias Lang, Marc Toussaint</i>	1263
Toward Human Interaction with Bio-Inspired Teams	
<i>Michael A. Goodrich, P. B. Sujit, Jacob W. Crandall</i>	1265
Escaping Heuristic Depressions in Real-Time Heuristic Search	
<i>Carlos Hernández, Jorge A. Baier</i>	1267
Pseudo-tree-based Algorithm for Approximate Distributed Constraint Optimization with Quality Bounds	
<i>Tenda Okimoto, Yongjoon Joe, Atsushi Iwasaki, Makoto Yokoo</i>	1269
Concise Characteristic Function Representations in Coalitional Games Based on Agent Types	
<i>Suguru Ueda, Makoto Kitaki, Atsushi Iwasaki, Makoto Yokoo</i>	1271
Iterative Game-theoretic Route Selection for Hostile Area Transit and Patrolling	
<i>Ondřej Vaněk, Michal Jakob, Viliam Lisý, Branislav Bošanský, Michal Pěchouček</i>	1273
Abduction Guided Query Relaxation	
<i>Samy Sá, João Alcântara</i>	1275
A Message Passing Approach To Multiagent Gaussian Inference for Dynamic Processes	
<i>Stefano Ermon, Carla Gomes, Bart Selman</i>	1277
Multiagent Environment Design in Human Computation	
<i>Chien-Ju Ho, Yen-Ling Kuo, Jane Yung-Jen Hsu</i>	1279
Social Distance Games	
<i>Simina Brânzei, Kate Larson</i>	1281

Agent Sensing with Stateful Resources	
<i>Adam Eck, Leen-Kiat Soh</i>	1283
Modeling Bounded Rationality of Agents During Interactions	
<i>Qing Guo, Piotr Gmytrasiewicz</i>	1285
Comparing Action-Query Strategies in Semi-Autonomous Agents	
<i>Robert Cohn, Edmund H. Durfee, Satinder Singh</i>	1287
A Multimodal End-of-Turn Prediction Model: Learning from Parasocial Consensus Sampling	
<i>Lixing Huang, Louis-Philippe Morency, Jonathan Gratch</i>	1289
Scalable Adaptive Serious Games using Agent Organizations	
<i>Joost Westra, Frank Dignum, Virginia Dignum</i>	1291
Integrating power and reserve trade in electricity networks	
<i>Nicolas Höning, Han Noot, Han La Poutré</i>	1293

Demonstrations

BDI Agent model Based Evacuation Simulation	
<i>Masaru Okaya, Tomoichi Takahashi</i>	1297
An Interactive Tool for Creating Multi-Agent Systems and Interactive Agent-based Games	
<i>Henrik Hautop Lund, Luigi Pagliarini</i>	1299
Towards Robot Incremental Learning Constraints from Comparative Demonstration	
<i>Rong Zhang, Shangfei Wang, Xiaoping Chen, Dong Yin, Shijia Chen, Min Cheng, Yanpeng Lv, Jianmin Ji, Dejian Wang, Peijia Shen</i>	1301
Teleworkbench: Validating Robot Programs from Simulation to Prototyping with Minirobots	
<i>A. Tanoto, F. Werner, U. Rückert, H. Li</i>	1303
A MAS Decision Support Tool for Water-Right Markets	
<i>Adriana Giret, Antonio Garrido, Juan A. Gimeno, Vicente Botti, Pablo Noriega</i>	1305
An Implementation of Basic Argumentation Components	
<i>Mikolaj Podlaszewski, Martin Caminada, Gabriella Pigozzi</i>	1307
AgentC: Agent-based System for Securing Maritime Transit	
<i>Michal Jakob, Ondřej Vaněk, Branislav Bošanský, Ondřej Hrstka, Michal Pěchouček</i>	1309
Bee-Inspired Foraging In An Embodied Swarm	
<i>Sjriek Alers, Daan Bloembergen, Daniel Hennes, Steven de Jong, Michael Kaisers, Nyree Lemmens, Karl Tuyls, Gerhard Weiss</i>	1311
The Social Ultimatum Game and Adaptive Agents	
<i>Yu-Han Chang, Rajiv Maheswaran</i>	1313
DipTools: Experimental Data Visualization Tool for the DipGame Testbed	
<i>Angela Fabregues, David López-Paz, Carles Sierra</i>	1315
TALOS: A Tool for Designing Security Applications with Mobile Patrolling Robots	
<i>Nicola Basilico, Nicola Gatti, Pietro Testa</i>	1317
Vision-Based Obstacle Run for Teams of Humanoid Robots	
<i>Jacky Baltes, Chi Tai Cheng, Jonathan Bagot</i>	1319
Evolutionary Design of Agent-based Simulation Experiments	
<i>James Decraene, Yew Ti Lee, Fanchao Zeng, Mahinthan Chandramohan, Yong Yong Cheng, Malcolm Yoke Hean Low</i>	1321
Interactive Storytelling with Temporal Planning	
<i>Julie Porteous, Jonathan Teutenberg, Fred Charles, Marc Cavazza</i>	1323
Agent-based Network Security Simulation	
<i>Dennis Grunewald, Marco Lützenberger, Joël Chinnow, Rainer Bye, Karsten Bsufka, Sahin Albayrak</i>	1325
Experimental Evaluation of Teamwork in Many-Robot Systems	
<i>Andrea D'Agostini, Daniele Calisi, Alberto Leo, Francesco Fedi, Luca Iocchi, Daniele Nardi</i>	1327

Doctoral Consortium Abstracts

Reasoning About Norms Within Uncertain Environments <i>Natalia Criado</i>	1331
Privacy and Self-disclosure in Multiagent Systems <i>Jose M. Such</i>	1333
Policies for Role Based Agents in Environments with Changing Ontologies <i>Fatih Tekbacak, Tugkan Tuglular, Oguz Dikenelli</i>	1335
Human Factors in Computer Decision-Making (PhD Thesis Extended Abstract) <i>Dimitrios Antos</i>	1337
Security in the Context of Multi-Agent Systems <i>Gideon D. Bibu</i>	1339
Agent Dialogues and Argumentation <i>Xiuyi Fan</i>	1341
Massively Multi-Agent Pathfinding made Tractable, Efficient, and with Completeness Guarantees <i>Ko-Hsin Cindy Wang</i>	1343
Securing Networks Using Game Theory: Algorithms and Applications <i>Manish Jain</i>	1345
Decentralized Semantic Service Discovery based on Homophily for Self-Adaptive Service-Oriented MAS <i>E. del Val</i>	1347
A Cost-Oriented Reorganization Reasoning for Multiagent Systems Organization Transitions <i>Juan M. Alberola</i>	1349
Graphical Multiagent Models <i>Quang Duong</i>	1351
Extended Abstract of Elisabetta Erriquez Thesis <i>Elisabetta Erriquez</i>	1353
Improving Game-tree Search by Incorporating Error Propagation and Social Orientations <i>Brandon Wilson</i>	1355
Negotiation Teams in Multiagent Systems <i>Víctor Sánchez-Anguix</i>	1357
Real-World Security Games: Toward Addressing Human Decision-Making Uncertainty <i>James Pita</i>	1359
A Multi-Agent System for Predicting Future Event Outcomes <i>Janyl Jumadinova</i>	1361
A Study of Computational and Human Strategies in Revelation Games <i>Peled Noam</i>	1363
Thesis Research Abstract: Modeling Crowd Behavior Based on Social Comparison Theory <i>Natalie Fridman</i>	1365
Cooperation between Self-Interested Agents in Normal Form Games <i>Steven Damer</i>	1367
Group Decision Making in Multiagent Systems with Abduction <i>Samy Sá</i>	1369
Security Games with Mobile Patrollers <i>Ondřej Vaněk</i>	1371
Self-Organization in Decentralized Agent Societies through Social Norms <i>Daniel Villatoro</i>	1373
A Trust Model for Supply Chain Management <i>Yasaman Haghpanah</i>	1375

Main Program – Best Papers

Best Papers Session I

Agent-Based Control for Decentralised Demand Side Management in the Smart Grid

Sarvapali D. Ramchurn, Perukrishnen Vytelingum, Alex Rogers, and Nick Jennings
Intelligence, Agents, Multimedia Group
School of Electronics and Computer Science
University of Southampton, UK
{sdr,pv,acr,nrj}@ecs.soton.ac.uk

ABSTRACT

Central to the vision of the smart grid is the deployment of smart meters that will allow autonomous software agents, representing the consumers, to optimise their use of devices and heating in the smart home while interacting with the grid. However, without some form of coordination, the population of agents may end up with overly-homogeneous optimised consumption patterns that may generate significant peaks in demand in the grid. These peaks, in turn, reduce the efficiency of the overall system, increase carbon emissions, and may even, in the worst case, cause blackouts. Hence, in this paper, we introduce a novel model of a Decentralised Demand Side Management (DDSM) mechanism that allows agents, by adapting the deferment of their loads based on grid prices, to coordinate in a decentralised manner. Specifically, using average UK consumption profiles for 26M homes, we demonstrate that, through an emergent coordination of the agents, the peak demand of domestic consumers in the grid can be reduced by up to 17% and carbon emissions by up to 6%. We also show that our DDSM mechanism is robust to the increasing electrification of heating in UK homes (i.e., it exhibits a similar efficiency).

Categories and Subject Descriptors

I.2.11 [Computing Methodologies]: Artificial Intelligence—*Distributed Artificial Intelligence*

General Terms

Agents, Multi-Agent Systems

Keywords

Energy, Demand-side Management Electricity, Multi-Agent Systems, Agent-Based Control, Agents.

1. INTRODUCTION

The creation of the Smart Grid has been posed as one of the greatest challenges of this century, as countries face dwindling non-renewable energy sources and the adverse effects

Cite as: Agent-Based Control for Decentralised Demand Side Management in the Smart Grid, S. D. Ramchurn, P. Vytelingum, A. Rogers, N. R. Jennings, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems – Innovative Applications Track (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 5-12.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

of climate change due to carbon emissions [12]. The vision of a Smart Grid includes technologies that enable the efficient integration of intermittent renewable energy sources (such as wind or solar energy) and electric vehicles, and will reduce demand by allowing consumers to better manage how electricity is used, stored, and delivered. One of the key underpinnings of this endeavour is the concept of the *smart meter* which aims to manage the devices in the home to minimise inefficiencies in usage and maximise the user's savings. Smart meters also aim to interact with the grid in order to help reduce peaks in demand (which would otherwise lead to instability in the grid, higher energy costs and higher carbon emissions) and keep up with variable output from wind or solar energy generators [1]. If these ideal features of smart metering materialise, they may lead to significant reductions in energy costs and carbon emissions while guaranteeing security of supply.

To date, smart meters have mainly been designed to act as information provisioning devices that tend to leave it to the user to manage devices in the home, with the hope they will reduce their energy demands. In addition to this, Demand Side Management (DSM) technologies have also been developed to alter the behaviour of users by either charging them for using electricity at peak hours (potentially leading to other peaks at cheaper hours) or by inducing the devices, via the smart meter, to turn off (or on) when the network signals them to (either through price signal or the frequency of AC power) [6, 7].¹ While such DSM techniques have been shown to bring about significant improvements on a small number of houses, it is unclear how such technologies will scale when smart meters are rolled out to millions of homes or buildings nationwide.² In particular, the centralised management of even thousands of smart meters is likely to be a complex task that may require intruding upon users' privacy to cater for all homes, each with its own specific set of devices, usage profile, and preferences set by the owner (e.g., thermostat settings or times at which she wishes to switch on energy intensive devices such as washing machines). Moreover, as we show in this paper, simply leaving smart meters to react to price or frequency fluctuations can cause all devices to respond *at the same time* (e.g., every user switching

¹A change in frequency is caused by unmatched supply and demand in the grid. Maintaining the frequency of electricity (50Hz in UK) is important because devices running on AC power such as electric motors in household appliances are optimised for this frequency and may be damaged if it deviates too far.

²As per the Climate Change Act of 2008, the UK Government has committed to installing smart meters in all of the 26M homes in the UK by 2020 [2].

on the air conditioning unit or washing machine at the same time), thus generating peaks in demand that impact on the system. Finally, the fact that increasingly more and more features of the home are likely to be electrified in the future [3] (e.g., the use of heat pumps for space and water heating), means that more significant peaks may be created due to the reactive behaviour of the smart meters. Thus, unless appropriate control strategies are implemented to allow smart meters to *coordinate*, they may well generate greater peaks in demand than before, leading to a more stressed grid and, in the worst case, blackouts.

Against this background, in this paper, we develop and evaluate a model of *decentralised demand side management* (DDSM) to coordinate large populations of autonomous agents representing individual smart meters. In more detail, we model the *smart home* as being composed of a number of deferrable loads (controlled by an agent that optimises the deferment of these loads so as to maximise the comfort in the home and minimise energy costs) as well as non-deferable loads. Moreover, we design mechanisms for agents to *adapt* (instead of reacting as in traditional DSM mechanisms) the deferment of their loads. Thus, this paper advances the state of the art in the following ways. First, we provide a model of the smart home where the deferment of loads is optimised using mathematical programming techniques. Second, we provide motivating examples that illustrate the key requirements to implement fully decentralised agent-based control strategies and thereon show that our model of DDSM meets these requirements. Third, we empirically evaluate our DDSM mechanism (on a population of 5000 agents) and show that using DDSM leads to the *emergent coordination* of agents (without directly communicating with each other). Based on the average load profile of 26M UK homes, the agents converge to an equilibrium where the peak demand of the domestic consumers is flattened by up to 17% and carbon emissions are reduced by up to 6%. Furthermore, using evolutionary game theoretic techniques, we also show that it is always profitable (i.e., it is a Nash Equilibrium) to allow agents to control the deferment of devices as the proportion of smart homes in the population increases. Finally, we demonstrate that our DDSM mechanism would remain effective even in a future with significantly more electrified heating.

The rest of this paper is structured as follows. In Section 2, we review the literature and in Section 3, we describe our model of the smart home. Section 4 presents the mathematical programming solutions we propose to the optimisation problems that may need to be solved by the smart meter. We present the requirements for implementing the DDSM and then, our DDSM in Section 5. In Section 6, we empirically evaluate the performance of the DDSM in terms of its impact on the smart home and the grid performance. Section 7 concludes.

2. BACKGROUND

Scheppe *et al.* introduced the concept of DDSM and were the first to propose a design for the smart meter which included advanced functions to optimise the schedule of loads, the prediction of demand in the home, and the prediction of weather conditions among others [10].³ While being far ahead of their time, their work mainly dealt with predicting

³Prior to this, they had provided techniques to (i) control demand and supply in the network through the use of spot pricing of electricity [11] (ii) perform demand side manage-

the system behaviour, given the implementation of smart meters, using closed form solutions that required approximating and homogenising the behaviours of actors involved in the system. In contrast, in this paper, we present an agent-based approach that allows us to model each individual home in the system and to simulate large numbers of such homes (potentially in the order of millions given enough computational resources) in order to analyse system performance. In so doing, we are able to establish the incentives of the users to adopt smart metering technology using agent-based simulations and evolutionary game theoretic (EGT) techniques.

Similar to our work is that of Vytelingum *et al.* [13] who presented a model of smart meters controlling storage devices in the home and showed how doing so improves the performance of the system at large. However, they do not deal with more complex deferrable loads and managing the comfort in the home, which are key issues in DSM. Hence, we believe our paper generalises their approach by formalising and evaluating the control mechanism needed to elicit good equilibria in the system when performing demand side management.

In addition to these theoretical studies, recent DSM trials by the GridWise Project⁴ have shown that market-based control techniques, where homes respond to real-time pricing (RTP), can reduce peak demand (when prices are high) to some degree [6, 7]. Theoretical studies of this setting also predict similar benefits [8]. However, these approaches simplify the behaviour of the devices in the home to *reactively* respond to prices rather than predicting and planning use. This uncoordinated behaviour, in turn, leads to peaks in demand being shifted to periods when the prices are low. In contrast, in our DDSM, the agents *adapt* the behaviour of the devices which allows them to coordinate, in an emergent fashion, to flatten demand.

3. THE SMART HOME MODEL

In this section we present a model of an agent that optimises the home's energy usage by managing loads to maximise savings while mitigating the impact on the user's lifestyle (or comfort). Now, it is important to note that loads can be classified into two categories: those that can and those that cannot be deferred to later times. Examples of the latter include lighting, entertainment devices, phone charging and computer usage and of the former, include washing machines, dishwashers, boilers and fridges. Hence, the agent is only able to control some of the devices in the home without impacting too much on the lifestyle of the user. Moreover in the UK, deferrable loads currently account for around 20% of the domestic electricity usage and this is likely to grow with the increased electrification of space and water heating [2, 9].⁵ Now, if agents are not coordinated in deferring these loads, they may induce higher peaks in the system than before (as discussed in Section 1). Thus, before proposing our solution to this, in the following subsections we build a model of a smart home that is sufficiently realistic to cap-

ment through market-based exchanges of energy rights to reduce peaks in electricity demand [15], and (iii) optimise the heating module of a building or a set of buildings through the use of spot pricing.

⁴See more details at <http://gridwise.pnl.gov/>.

⁵Mackay convincingly argues that heat pumps are much more efficient than (micro) combined heating and power (CHP) technology that is currently more popular.

ture the key effects that we address. In the next subsection, we elaborate on the types of devices we consider.

3.1 Deferrable Loads

Within the domestic energy domain, it is common to characterise the devices under four specific categories [5]: (i) wet (e.g., washing machine or dishwasher), (ii) cold (e.g., fridge or freezer), (iii) water heating (e.g., boiler or hot water dispenser) (iv) space heating (e.g., heat pumps or radiators). The devices that fall under these different categories behave very differently. For example, the wet types usually involve set periods of time at which they are switched on and off either by the user or by the device controller. The cold, water and space heating devices are much more dependent on the usage and the temperature of the home and will vary their behaviour depending on various external factors beyond the control of the user. Hence, we categorise these different loads in terms of those that have precise running times and running periods, which we call *shiftable static loads* (SSLs), and those that are more dependent on the temperature, which we call *thermal loads*.⁶ Given this, in the next sub-sections, we provide exemplar optimisation models for SSLs and a thermal load. We will consider a typical domestic profile that spans the usage over a day divided in half-hourly slots represented by a set of time slots $t \in T$ where $T = \{1, \dots, 48\}$. We also assume the smart meter receives a price signal (which may be fixed) at every time t .

3.2 Shiftable Static Loads

The set of SSLs is noted as $l \in L$ where each load is characterised by a set of parameters. Thus, $o_l \subseteq T$ is a set of time slots at which the device is set by the user to turn on. In effect, o_l represents the preferred time at which the user would like to switch on the device (e.g., switch on the washing machine while she is away). Any deviation from this preset time is likely to cause a loss of comfort to the user. Let $d_l \in \{-24, -23, \dots, 23, 24\}$ be the deferment of the device and note $r_l \in \mathbb{R}^+$ as the power rating of the device in kW. The duration v^l (in time slots) of each load is drawn from a uniform distribution, $U(v_{min}^l, v_{max}^l)$ where $v_{min}^l, v_{max}^l \in T$ will depend on the device used (e.g., a washing machine typically runs between 1 and 3 hours). To model the effect of shifting loads from their preset time, let $\Delta c_l \in \mathbb{R}^+$ be the *marginal comfort cost* associated with deferring the device to a different time from those initially set in o_l . Then, the overall comfort cost is $c_l = \Delta c_l |d_l|$. This assumes that the more the smart meter deviates from the time at which the user initially sets a device to run, the more discomfort it will cause to the user.⁷ The key variable in our optimisation model is effectively d_l since it determines the power consumption (which, in turn, determines the price paid by the user) and comfort cost of the user. Given a price p_t (in £/kWh), $\forall t \in T$, the objective is then to minimise the cost and comfort cost, as follows:

$$\begin{aligned} & \min_{d_l \forall l \in L} \sum_{l \in L} \sigma c_l + \sum_{t \in T} \sum_{l \in L} \gamma_t^l r_l p_t v^l 0 \\ & \text{such that } \gamma_t^l = \begin{cases} 1 & , t = t' + d_l \\ 0 & , \text{otherwise} \end{cases}, \quad (1) \\ & \text{where } t' \in o_l \end{aligned}$$

⁶It is possible to cast cold loads such as small fridges under SSLs if, for example, it is only required for the fridge to keep its contents well below room temperature (instead of a set temperature) by turning it on and off at regular intervals).

⁷Of course, more complex functions could be built by the user, but this is beyond the scope of this paper.

such that $\gamma_t^l \in \{0, 1\}$ determines when the devices is turned on or off (based on the shifting of the on times o_l by d_l), v_l is a number of time slots over which device l is on, and $\sigma \in [0, 1]$ is a scaling factor determining how much comfort is more important to the user relative to price. Using the above mathematical programming formulation, the optimisation problem of finding the optimal deferment of SSLs be therefore be directly solved using standard solvers.

3.3 Thermal Loads

We now turn to modelling the thermal properties of the home and the use of a heater controlled by an *intelligent* adaptive thermostat controlled by the smart meter.⁸ Standard adaptive thermostats warm up the house to be at the required temperature exactly when the user expects to be home. By turning on the heating before the user is present, the thermostat ensures the house is always warm when the user requires it to be. An intelligent adaptive thermostat, in turn, aims to optimise the heating (or cooling) profile (i.e., times at which the heater is turned on), in order to guarantee the required temperature is reached at the time the user is present *and* that the costs of doing so are minimised. To this end, we specify the properties of our heating optimisation model as follows. First, we detail the properties of the home and the heater. The heater we choose to model is a heat pump (either air-source or ground-source) which simply extracts heat from one place and moves it to another place [9]. Second, we provide a mixed-integer quadratic programming (MIQP) formulation of the problem which can be solved using off-the-shelf solvers.⁹

3.3.1 Home and Heater Properties

To model and simulate the thermal properties of the home and heater, we build upon the models proposed by [4] which use the ASHRAE model of comfort.¹⁰ Let $\phi \in \mathbb{R}^+$ be the thermal leakage rate of the house measured in W/K. The internal temperature of the home at time t is $\tau_{in}^t \in \mathbb{R}^+$ and the external temperature (in K) is denoted as $\tau_{ext}^t \in \mathbb{R}^+$. The user will set the temperature $\tau_{opt} \in \mathbb{R}^+$ at the level she feels comfortable (typically 22.5 degrees Celsius). The heat capacity and the total mass of air in the home are denoted as $a_{hc} \in \mathbb{R}^+$ (in J/kg/K) and $a_{mass} \in \mathbb{R}^+$ (in kg) (which is computed using the air density and house volume) respectively. The heater is described by r_h and o_h where $r_h \in \mathbb{R}^+$ is the power rating (in kW) of the heater and $o_h \subseteq T$ is a set of time slots at which the heat pump is switched on. Given this, we also define the variable $h_{on}^t \in \{0, 1\}$ for every $t \in T$ where $h_{on}^t = 1$ if $t \in o_h$ and 0 otherwise. This description of the heater is similar to the deferrable loads above except that the deferment of the heater is much more complex as we will see next. For now, we can compute the total heat input in the system as:

$$\eta_i^t = h_{on}^{t-1} r_h - \phi(\tau_{in}^{t-1} - \tau_{ext}^{t-1}) \quad (2)$$

Moreover, we can define the relationship between the inside temperature at time t and the amount of heat injected in

⁸The model also applies to cooling effects simply by setting the parameters differently to model heat extraction rather than heat injection as we do here.

⁹We use IBM ILOG CPLEX 12.2 in our experiments.

¹⁰American Society of Heating, Refrigerating and Air-Conditioning Engineers (ASHRAE) Standard 55-2010: Thermal environmental conditions for human occupancy.

the house at time $t - 1$ as:

$$\tau_{in}^t = \tau_{in}^{t-1} + \frac{k\eta_i^t}{a_{hc}a_{mass}} \quad (3)$$

where $k = \frac{24 \times 3600}{|T|}$ is the time (in seconds) during which the heater is on.¹¹

Now, we expect users to set the times at which they will be at home and will want the temperature adjusted for their comfort. To this end, we define comfort on times as a $C_{on}^t \in \{0, 1\}$ at every time slot $t \in T$ such that the $C_{on}^t = 1$ if the user needs comfort and $C_{on}^t = 0$ otherwise. During every time slot, we model the *instantaneous* comfort cost (i.e., ignoring changes from the previous time slot) of the user due to heating as $\Delta c_h^t \in \mathbb{R}^+$ such that:

$$\Delta c_h^t = \begin{cases} C_{on}^t \omega_1 (\tau_{in}^t - \tau_{opt})^2, & \tau_{in}^t \geq \tau_{opt} \\ C_{on}^t \omega_2 (\tau_{in}^t - \tau_{opt})^2, & \tau_{in}^t < \tau_{opt} \end{cases} \quad (4)$$

where $\omega_1, \omega_2 \in [0, 1]$ are constants that scale the effect of the temperature difference depending on whether it is more (or less) comfortable when the temperature is higher or lower than t_{opt} (in degrees celcius). Typically, in colder periods, $\omega_1 < \omega_2$ is preferred. Then, the total comfort cost at time t is a combination of the present instantaneous comfort cost and at $t - 1$ given by:

$$c_h^t = \Delta c_h^t + \gamma \Delta c_h^{t-1} \quad (5)$$

where $\gamma \in [0, 1]$ scales the effect of the previous time slot on the current one (and captures the psychological persistence of discomfort).

3.3.2 MIQP Formulation

Given the times when $C_{on}^t = 1$ and the comfort cost c_h^t , the goal of the agent is to optimise the heating so as to minimise the price paid by the user. Thus, assuming the price of electricity at every time t is p_t , and specifying $h_{on}^t \in \{0, 1\}, \forall t \in T$ as the decision variables, the objective function of the program is:

$$\min_{h_{on}^t, \forall t \in T} \sum_{t \in T} c_h^t + \kappa (v^h h_{on}^t r_h p_t) \quad (6)$$

subject to (2),(3), and (4), where $\kappa \in [0, 1]$ balances the cost of heating against the comfort and $v^h = 1800s$ is the duration of one time slot in seconds (assuming each time the heater is turned on for a half-hour). The quadratic nature of the objective function arises as a result of the computation of the comfort cost in (4). We set κ to a very small value to ensure that this part of the objective function only ensures that the user mainly optimises her comfort. Now, in some cases, a budget can be set by the user according to how much she can afford to spend on a daily basis for heating purposes. Thus, in order to ensure that the cost is never higher than a set value $p' \in \mathbb{R}^+$, we also add the following constraint to the model:

$$\sum_{t \in T} v^h h_{on}^t r_h p_t \leq p' \quad (7)$$

Since agents are only intent on maximising their gains (in comfort and cost savings), their aggregated uncoordinated behaviour may result in poor system performance unless demand-side management mechanisms are put in place.

¹¹In our experiments, we refine the resolution to smaller time slots (e.g., of 5 or 10 minutes) to get a better measure of the temperature inside the room.

Hence, given our model of the smart home, in the next section, we first describe the optimal and centralised solution that maximises social welfare (and describes what optimal means in this context) that determines the optimal coordinated behaviour of the agents. By so doing, we can use this as a benchmark for any control mechanism that can be developed in this domain (which we use to evaluate our mechanism in Section 6).

4. MAXIMISING SOCIAL WELFARE

As a result of the individual agents' deferment of loads, the consumption of energy at different times of the day may significantly increase or decrease, resulting in price spikes in the system. Before going on to design a control mechanism to reduce this effect, however, we need to determine the optimal performance of the population of agents as a whole to evaluate our control mechanism (see Section 6).

As was shown in [13], when agents attempt to defer consumption (in their case, they used storage), a game theoretic analysis of the problem predicts that the agents should converge to the competitive equilibrium. This means that they converge to an equilibrium (flat) price for electricity if there is enough storage, at which point social welfare is maximised.

In the context of deferrable loads, such an analysis is significantly more complex as shiftable static loads, comfort costs, and thermal loads are not homogeneous across the population of agents. Instead, we can compute the point at which all agents behave optimally by aligning their individual objective functions with the global optimum as follows. First, we assume that each agent $i \in I$, where I is the set of agents, has a non-shiftable static load l_{fixed} such as lighting, cooking and other types of essential loads which have a power rating $fixed_{t,i} \in \mathbb{R}^+$ in kW at every time point $t \in T$. Second, we assume that the cost of electricity for the population is given by a quadratic cost function (the function is assumed to be quadratic to mimic an increasing marginal cost for electricity supply – we use such a model in our evaluation) $s_t : \mathbb{R}^+ \rightarrow \mathbb{R}^+$ for every time slot t such as $s_t(q) = \theta q^2$ where $\theta > 0$. This means that the greater the demand from the agents, the higher will be the unit price of electricity (because of the monotonically increasing function). Then, we aggregate the functions in equations (1) and (6) subject to the individual constraints as stated in Equations (2), (3) and (5) and the conditions stated in Sections 3.2 and 3.3.1 as follows:

$$\min_{h_{on,i}^t, \forall t \in T, d_i^t, \forall i \in I} \sum_{i \in I} \left(\begin{aligned} & \sum_{t \in T} c_{h,i}^t + \sum_{l \in L_i} \sigma c_{l,i} \\ & + \sum_{t \in T} s_t \left(\sum_{l \in L_i} \gamma_{t,i}^l r_{l,i} v_i^l \right) \\ & + \kappa_i v_i^h h_{on,i}^t r_{h,i} + fixed_{t,i} \end{aligned} \right) \quad (8)$$

The main difference between the above objective and the individual agents' objective is that the cost of electricity in the above case is based on the induced cost as a result of the agents' aggregated demand (both from their deferrable loads and their static demand) using the supply function $s(\cdot)$ rather than the cost predicted by the agents p_t (which is assumed to be a fixed unit cost).

Note that the optimal algorithm above is unlikely to scale very well in the number of loads per agent and deferability of these loads given the complexity of the optimisation problem and given non-linearity of the objective function. Specifically, experiments (where Equation (4) is linearised) using a standard MIQP solver (e.g., IBM ILOG CPLEX

12.2) could be run for up to 75 agents with up to three deferrable loads each (on a 64-bit machine with 12GB of RAM). Beyond this size, the solver cannot model the problem in memory. This would be fine for scenarios where a central controller has complete control over the few deferrable loads in a building for example, it will clearly not scale to hundreds of houses (where users might not want the central controller to override their preferences to use electricity) or for deferrable loads (and certainly not to all 26M UK homes).

Hence, in the next section, we discuss some of main issues in DSM and introduce a novel approach to tackling this problem in a completely decentralised fashion.

5. DECENTRALISED DEMAND-SIDE MANAGEMENT

Most DSM approaches involve a central controller that advises a pool of consumers to reduce their current demand (during peak demands), often by reducing or deferring loads subject to economic incentives. This approach has repeatedly been shown to be effective for relatively small pool sizes of industrial and commercial consumers [6]. Indeed, it constitutes the business model of a number of energy service companies (ESCOs) that exclusively deal with Demand Side Management (e.g., EnergyConnect Inc. and EnerNOC). While it remains feasible to signal a small number of consumers and expect an immediate response, DSM at a regional or national level (with 26M of households in the UK) is more complex. Given this, in the next subsections, we discuss the main issues associated with applying DSM on a large scale (dealing with thousands and millions of homes). First, we justify the need for a price signal to incentivise the agents to defer their demand. Second, we show that, even if an accurate price signal is provided according to the criteria we propose, the adaptive and autonomous behaviour of the agents in the system is the key component that can enable significant performance benefits in the Smart Grid. Thus, we propose a novel model of decentralised demand side management (DDSM).

5.1 The Pricing Mechanism

The pricing mechanism that we propose involves signalling the costs of generating electricity to the consumers. Traditionally, consumers are offered a fixed price for electricity by their supplier. Exceptions to this include time-of-use (TOU) pricing (e.g., Economy 7 heating or eco:2020 in the UK) and real-time pricing (RTP) schemes. While fixed pricing mechanisms completely hide the real-time costs of electricity (which varies according to the type of generators used and availability of intermittent renewable energy), TOU pricing simply *bias*es the real price of electricity in order to incentivise users (who typically aim to maximise their savings) to shift their loads to off-peak periods (i.e., when aggregate demand is lower). In contrast, RTP involves providing a price signal for the next 30 minutes time slot at the current time. While RTP is still being evaluated in a number of trials, it has been shown to be better than TOU pricing in allowing users to dynamically adjust their demand to avoid peaks when electricity is more expensive [6, 7].

To understand the effect of the different pricing schemes on demand, consider Figure 1 where we show how 500 smart homes¹² (as described earlier) would react to the three dif-

¹²We implemented each smart home according to the settings

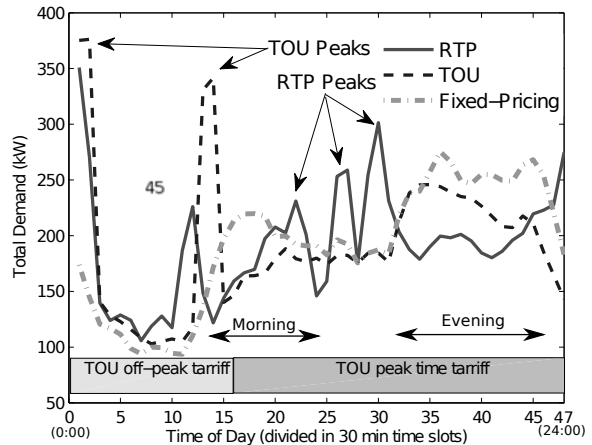


Figure 1: Demand optimisation given the Domestic Economy 7 tariff and the 30-min-tariff pricing models.

ferent types of pricing mechanisms (where real time prices are provided for 30-minute time slots). As expected, a fixed price induces a relatively high demand with large plateaus in the morning and evening (when users are at home and typically use their loads) since users are indifferent about the real-time costs of generating electricity. More importantly, we can observe that when all the consumers are on a two-tariff mechanism, they optimise their demand at the same time (to take opportunity of off-peak prices), and thus the ensuing demand on the grid now peaks during off-peak times. Similarly, it can be seen that the RTP mechanism also shifts demand to different times of the day and induces other peaks at these times.

These results show that a more accurate signal (i.e., representing real costs) allows consumers to better optimise their demand by reacting more often to a more accurate 30-min-tariff pricing model. However, it is also clear that the demand *cannot be flattened* by applying *only* a RTP mechanism, while completely ignoring the behaviour of the agents. This is because if the agents are signalled a low price for the next 30-minute period, they will all switch on their devices, which then results in a peak in demand at the next time period. When such a mechanism is rolled out on a large scale, such *reactive* behaviours can cause significant peaks. To remedy this, in the next sub-section, we propose a novel adaptive behaviour for the agents, which builds upon the RTP mechanism, to allow agents, *without any centralised control*, to coordinate in our DDSM model.

5.2 The Adaptive Mechanism

Our adaptive mechanism for DDSM is composed of algorithms that determine how to defer the two different types of deferrable loads present in the home (i.e., shiftable static loads and thermal loads) and how to optimise heating. First, we adopt the Widrow-Hoff learning mechanism [13] to gradually adapt how agents defer their deferrable loads based on predicted market prices for the *next day*¹³ (as opposed to the next 30 minutes as in [6, 7]). Specifically, an agent i gradu-

given in the evaluation section.

¹³We use a weighted moving average to predict day-ahead market prices, but more sophisticated approaches incorporating domain-specific knowledge could easily be used.

ally adapts its deferment parameter d_i^i towards the optimal $d_i^{i,*}$ (as computed by the objective function in Equation (1)) as follows:

$$d_i^i(t+1) = d_i^i(t) + \beta^i(d_i^{i,*} - d_i^i(t))$$

where $\beta^i \in (0, 1]$ defines the learning rate (i.e. how fast the agent reacts to changing conditions).

Second, each agent’s behaviour is further modelled by how often it readjusts its heating profile. Specifically, it reoptimises its thermal load profile on any particular day, with a probability of $\alpha \in (0, 1)$. This means, it reruns the optimisation detailed in Section 3.3.2 with a probability of α according to the predicted prices p_t for the next day. The expected number of agents that will optimise on any particular day is thus given by $N\alpha$, where N is the number of agents with DSM capability. By so doing, we introduce inertia into the optimisation by ensuring that all DSM-capable agents do not reoptimise at the same time and, in the next section, we empirically demonstrate that a stable and desirable outcome is reached whereby grid demand flattens when α and β are sufficiently small. Moreover, we also show that the agents’ best strategy (profit maximising) is to adopt an adaptive behaviour. Thus, even though consumers are not centrally coordinated for DSM, our mechanism ensures an effective *emergent* coordination among consumers.

In general, our DDSM mechanism can be particularly attractive when dealing with millions of consumers since it does not require, apart from the price signal (which could be directly sent to the smart meter), any other form of direct communication between the generators and the consumers. Moreover, the adaptive mechanism can also be modelled for different types of consumers as discussed above. Thus, when large populations of such consumers are simulated within a smart grid, it is possible, with reasonable accuracy, to predict the behaviour of the system. To this end, in the next section, we provide simulations of populations of smart homes (extrapolated across the grid¹⁴) and evaluate the performance of our DDSM both in terms of the consumers’ benefits and the grid performance. In so doing, we aim to show how our DDSM mechanism generally aligns the incentives of the users with the system-wide objectives, leading to a more efficient and stable Smart Grid.

6. EMPIRICAL EVALUATION

In our experiments we consider a population of 5000 agents¹⁵ and a RTP pricing based on the macro-model of the UK electricity market (as outlined by Vytelingum *et al.* [13]), with a quadratically increasing marginal cost. We used, for the shiftable static loads and thermal loads, data compiled by the Department of Trade and Industry in the UK from 2008.¹⁶ This dataset shows that, while shiftable static loads

¹⁴We use a similar methodology to [13] based on real UK Grid demand and market prices and average UK domestic load profiles for a typical weekday in winter.

¹⁵The time taken to run our simulations grows linearly with the number of agents and we have confirmed similar results with populations of up to 10,000 agents. However, when we compare to the optimal solution, we are limited to 75 agents, due to the computational complexity of the solution. Hence, to approximate the optimal solution for 5000 agents, we ran the global optimisation in Equation (8) 100 times for numbers of agents from 50 to 75 and estimated the optimal solution for 5000 agents from the trend identified.

¹⁶<http://www.berr.gov.uk/files/file11250.pdf> on 20/10/2010.

(e.g., wet loads such as washing machines and dryers) are owned by nearly the whole population, the penetration of electrical heating is currently only around 7%. Moreover, shiftable static loads take up on average 20% of the total energy usage of a house while the rest of the consumption is due to entertainment, lighting, and cooking purposes. This setup is likely to change in the future because of a growing need to use more efficient heating devices such as heat pumps such that the load profile is likely to change significantly.

Given this, in this section, we also aim to show the effects of increased deferment of electrified thermal load, on the smart grid. To this end, we create instances of the smart home with parameters set according the dataset above. We model two wet shiftable static loads (dryer and washing machine) with power rating of 2kW each as well as a thermal load (with $\tau_{opt} = 22.5$, $a_c = 1000$, $a_{mass} \in [1000, 2000]$, and $r_h = 2kW$). In order to generate the specified ON times for the shiftable static loads, we use the average load profile as a probability density function (i.e., describing the probability of the device being set to be switched on by the user at each half-hour period) and use a Poisson distribution $Pois(\lambda)$ to generate the number of times the devices are switched on in a day (with $\lambda = 2$). Moreover, we generate τ_{ext} as the average 5 minutes temperature readings taken over a period of 5 days from the roof-top sensor at one of our university buildings in January 2010 (during the cold season). Wherever needed, we adjust the number of agents having electrified heating (i.e., starting with the current 7%). Our experiments are repeated 100 times and the results averaged and error bars plotted to represent the 95% confidence intervals.

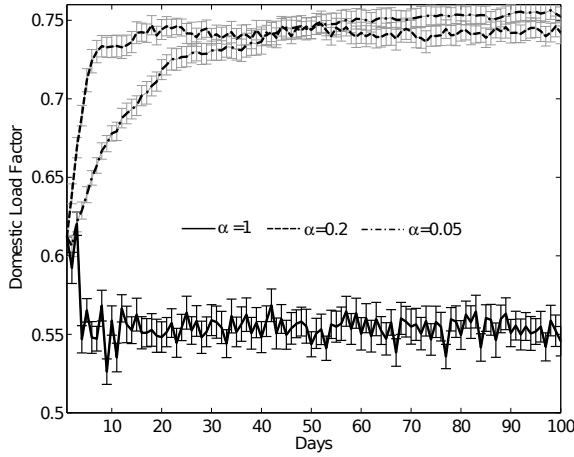
6.1 Performance of the Adaptive Mechanism

First, we evaluate the efficiency of our mechanism. We do so by comparing the efficiency of the grid when using our DDSM mechanism as opposed to a centralised system with complete and perfect information against an optimal behaviour. Given our optimal solution (described in Section 4) with an optimal load factor¹⁷ (LF) of 0.76, we can observe that a DDSM-based grid behaviour converges to the optimal behaviour (up from LF=0.6) when α is reasonably small as shown in Figure 2(a). While higher values may give faster convergence to the optimal, they do not allow the system to settle at a more efficient equilibrium (i.e. closer to the optimal). Furthermore, when α is too large, there is no convergence as too many agents are reoptimising at the same time such that the peaks are simply moved rather than flattened. When $\alpha = 1$ and every user optimises at the same time, the system breaks down (with an LF averaging 0.55). Thus, as we empirically showed, the system converges to the optimal behaviour without any centralised coordination when α is reasonably small. Specifically, for the rest of our experiments, we set $\alpha = 0.05$.

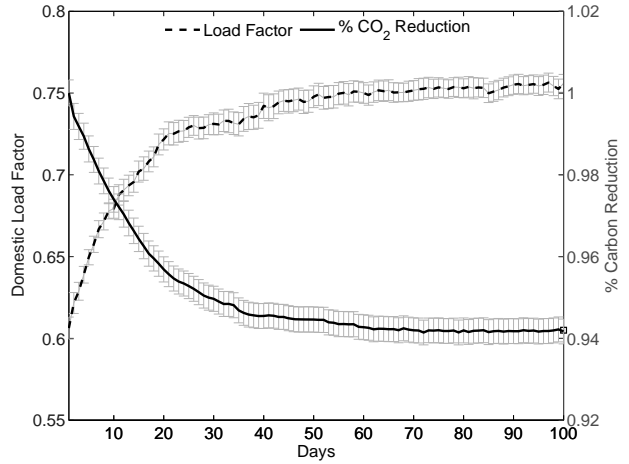
6.2 Emergent Behaviour of the Smart Grid

Given our DDSM approach with its adaptive mechanism, we first empirically demonstrate in Figure 2(b) that as the load factor of domestic consumers converges to the optimal load factor at 0.76 (meaning fewer peaks in the system), the

¹⁷The load factor is the ratio of average power to peak power and is ideally 1. A low load factor suggests large peaks in demand. Here we computed the load factor based on our estimation of the optimal solution as discussed in the previous section. We validated our results on small numbers of agents (50-75).



(a)



(b)

Figure 2: Evaluating DDSM in the Smart Grid w.r.t. (a) learning rates, (b) Load Factor & CO₂ emissions.

percentage of carbon reduction also increases up to 6% (i.e., electricity is produced from less polluting sources) as fewer carbon intensive peaking plants are used.

Next, we analyse the system by considering the load duration curve (LDC) of the grid. The LDC¹⁸ is used to illustrate the relationship between generating capacity requirements and capacity utilization, normalised to the current domestic load demand in UK. Figure 3 shows the LDC of the system (based on today's 7% electrification of heating) and, specifically, we can observe that particularly within the peak-load region (which is from 100% to 80%), the curve flattens when optimised, which implies a flattening of demand peaks. Furthermore, at 100% peak demand, we observe that the domestic load decreases by up to 17%, compared to the unoptimised case, which implies that the grid requires 17% less capacity to cope with domestic demand. Considering that current domestic peak demand is in excess of 15 GW, a 17% reduction in capacity requirement is significant and will become increasingly vital given the high rate of increase of domestic peak demand.

Next, we analyse the population dynamics to evaluate whether users will be incentivised to automate their smart meters (i.e., to give control to an agent). Specifically, we use evolutionary game theory (EGT) techniques which allow us to determine whether different proportions of the population will adopt smart meters (or not) depending on how much they can save by doing so. First, we formulate the problem as a complex non-deterministic game where agents have a mixed strategy $x_r \in (0, 1)$, i.e. a probability that they have DDSM capability and are only motivated by financial gains, where $r \in S = \{DDSM, \neg DDSM\}$. By analysing how x_r changes as the payoffs of agents with and without DDSM change for different x_r (assuming agents are more likely to

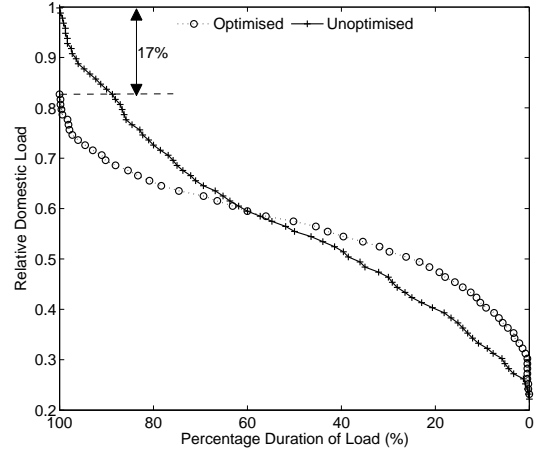


Figure 3: Load Duration Curve of Domestic Consumers with today's 7% electrification.

adopt the better strategy – whether or not to adopt smart metering), we aim to analyse how the proportion of the population adopting smart meters and DDSM evolves using the following equations [14]:

$$\dot{x}_r = [u(e^r, x) - u(x, x)]x_r \text{ where } u(x, x) = \sum_{r \in S} u(e^r, x)x_r$$

$$x_{nash} = \arg \min_{x \in (0,1)} \sum_{r \in S} (\max[u(e^r, x) - u(x, x), 0])^2$$

where $u(x, x)$ is the expected payoff of any agent (whether using DDSM or not), $u(e^r, x)$ is the expected payoff of an agent with DDSM given a proportion of the population with DDSM of x_r and, finally, x_{nash} is the Nash Equilibrium of the system (i.e., x_r where there are no incentive for an agent to deviate from). Figure 4 shows the payoffs of agents with and without DDSM. Based on our EGT analysis, we deduce that there is always an incentive for an agent to adopt DDSM given the higher payoffs for any x_r , such that \dot{x}_r is always positive. The population dynamics eventually converges to the Nash Equilibrium, $x_{nash} = 1$. This implies that all agents eventually adopt smart meters and DDSM.

¹⁸The load duration curve is computed as follows. Let ℓ be a vector of load values of the system at different time slots and ℓ^* be the ordered version (from high to low) of ℓ . The load duration curve is then given as $(f(x) \in X = \{\ell_1^*, \dots, \ell_M^*\} : x \in T^+ = \{T_1^+, \dots, T_M^+\})$, where $T_i^+ = \sum_{j=i}^M t_j$ and t_j is the time slot of the j^{th} ordered load value. The left-hand side represents the peak loads of the system. The LDC is a useful way of breaking down the load factor value to describe how long peaks last for. The domestic capacity (peak load) is given at 100% load duration.

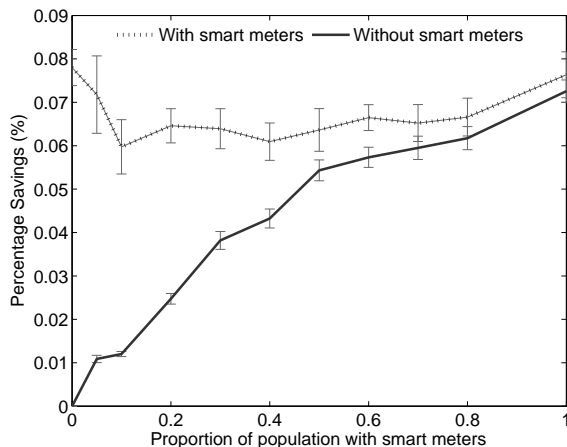


Figure 4: Percentage savings of agents with and without DDSM for different proportions of the population adopting smart meters and DDSM.

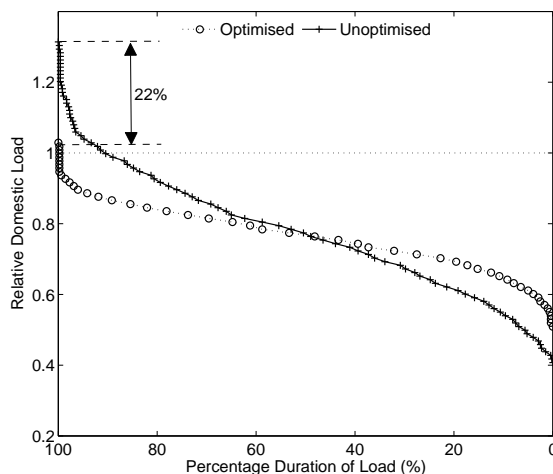


Figure 5: Load Duration Curve of Domestic Consumers with 25% electrification.

6.3 The Effect of Electrical Heating

Finally, we analyse future scenarios where the proportion of smart homes with thermal loads might increase. Specifically, Figure 5 shows grid performance for a potential future of 25% electrification. Thus, if the number of smart homes with thermal loads were to increase from 7% to 25%, the demands would be much higher (by up to 31% – based on initial loads from Figure 3), with higher peaks. Using DDSM, we demonstrate that these peaks would again be flattened significantly (shown by the significant drop in the 100% to 80%-peak region) using our novel model of a DDSM, with a 22% decrease in peak domestic capacity.

7. CONCLUSIONS

In this paper, we have provided a model of a smart home and presented our DDSM; a novel paradigm for mechanisms to manage demand on a large scale in the smart grid. Through our simulations involving 5000 homes and using average (winter) load profiles for 26M homes in the UK, we have shown that our DDSM mechanism can improve grid per-

formance by reducing peaks in demand by up to 17% and carbon emissions by up to 6%. We have also predicted, using evolutionary game theoretic techniques, that consumers will have significant economic incentives to adopt agent-based smart meters and will eventually all do so. Finally, we have shown that DDSM also reduces peaks as demand grows as a result of increased electrified heating. Future work will look at integrating price and weather predictions for more effective DDSM mechanisms and extending these mechanisms for commercial and industrial consumers in the smart grid. We also aim to evaluate our mechanism in scenarios where not all buildings are equipped with agents that can optimise their behaviour and remain insensitive to real-time prices as it is at present.

Acknowledgments

The work presented in this paper is funded through the IDEAS project (<http://www.ideasproject.info>) by a company active within the energy domain.

8. REFERENCES

- [1] Department of Energy and Climate Change. Smarter grids: The opportunity. HM Government, 2009.
- [2] Department of Energy and Climate Change. The UK low carbon transition plan: National strategy for climate and energy. Technical report, UK Dept. of E & CC, 2009.
- [3] R. Galvin and K. Yeager. *Perfect Power: How the MicroGrid Revolution Will Unleash Cleaner, Greener, More Abundant Energy*. McGraw-Hill Professional, 2008.
- [4] Y. Guo, R. Li, G. Poulton, and A. Zeman. A Simulator for Self-Adaptive Energy Demand Management. In *Proc. of the 2nd IEEE Conf. on Self-Adaptive and Self-Organizing Systems*, pages 64–73, 2008.
- [5] V. Hamidi, F. Li, and F. Robinson. Demand response in the uk domestic sector. *Electric Power Systems Research*, 79(12):1722 – 1726, 2009.
- [6] D. Hammerstrom et al. Pacific Northwest GridWise Testbed Demonstration Projects; Part II. Grid Friendly Appliance Project. Technical report, PNNL-17079, Pacific Northwest National Laboratory, 2007.
- [7] D. Hammerstrom et al. Pacific Northwest GridWise Testbed Demonstration Projects; Part I. Olympic Peninsula Project. Technical report, PNNL-17167, Pacific Northwest National Laboratory, 2008.
- [8] D. G. Infield, J. Short, C. Home, and L. L. Freris. Potential for domestic dynamic demand-side management in the uk. In *Proc. of the IEEE Power Engineering Society General Meeting*, pages 1–6, 2007.
- [9] D. MacKay. *Sustainable energy without the hot air*. UIT, Cambridge, 2009.
- [10] F. Scheppe, B. Daryanian, and R. Tabors. Algorithms for a spot price responding residential load controller. *IEEE Power Engineering Review*, 9(5):49–50, 1989.
- [11] F. Scheppe, R. Tabors, J. Kirtley, H. Outhred, F. Pickel, and A. Cox. Homeostatic utility control. *IEEE Transactions on Power Apparatus and Systems*, 99(3):1151 – 1163, 1980.
- [12] US Department of Energy. Grid 2030: A National Vision For Electricity's Second 100 Years, 2003.
- [13] P. Vytelingum, T. D. Voice, S. D. Ramchurn, A. Rogers, and N. R. Jennings. Agent-based micro-storage management for the smart grid. In *Proc. of the 8th Conf. Autonomous Agents And MultiAgent Systems, AAMAS 2010*, pages 39–46, 2010.
- [14] J. W. Weibull. *Evolutionary Game Theory*. MIT Press, Cambridge, MA, 1995.
- [15] R. D. Williams and F. C. Scheppe. Peak-power-demand limitation through independent consumer coordination. *IEEE Transactions on Systems, Man and Cybernetics*, 16(4):560 – 569, 1986.

Deploying Power Grid-Integrated Electric Vehicles as a Multi-Agent System

Sachin Kamboj, Willett Kempton
Center for Carbon-Free Power Integration
University of Delaware
Newark, DE 19716
{skamboj, willett}@udel.edu

Keith S. Decker
Dept. of Computer and Information Sciences
University of Delaware
Newark, DE 19716
decker@cis.udel.edu

ABSTRACT

Grid-Integrated Vehicles (GIVs) are plug-in Electric Drive Vehicles (EDVs) with power-management and other controls that allow them to respond to external commands sent by power-grid operators, or their affiliates, when parked and plugged-in to the grid. At a bare minimum, such GIVs should respond to demand-management commands or pricing signals to delay, reduce or switch-off the rate of charging when the demand for electricity is high. In more advanced cases, these GIVs might sell both power and storage capacity back to the grid in any of the several electric power markets — a concept known as Vehicle-to-Grid power or V2G power.

Although individual EDVs control too little power to sell in the market at an individual level, a large group of EDVs may form an aggregate or coalition that controls enough power to meaningfully sell, at a profit, in these markets. The profits made by such a coalition can then be used by the coalition members to offset the costs of the electric vehicles and batteries themselves. In this paper we describe an implemented and deployed multi-agent system that is used to integrate EDVs into the electricity grid managed by PJM, the largest transmission service operator in the world. We provide a brief introduction to GIVs and the various power markets and discuss why multi-agent systems are a good match for this application.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent systems

General Terms

Algorithms, Design, Performance, Experimentation

Keywords

Coalition formation, Vehicle-To-Grid, Grid-Integrated-Vehicle, Power Regulation

1. INTRODUCTION

Cite as: Deploying Power Grid-Integrated Electric Vehicles as a Multi-Agent System, Sachin Kamboj, Willett Kempton and Keith S. Decker, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems – Innovative Applications Track (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 13-20.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Plug-in Electric Drive Vehicles (EDVs), i.e. vehicles that use electricity to power at least part of their drivetrains, are becoming increasingly popular and offer some distinct advantages over their gasoline counterparts: they are cheaper to drive per-mile, produce fewer tailpipe emissions and have lower maintenance costs when compared to conventional gasoline vehicles. Grid-Integrated Vehicles (GIVs) are plug-in Electric Drive Vehicles (EDVs) with power-management and other controls that allow them to respond to external commands sent by power-grid operators, or their affiliates, when parked and plugged-in to the grid. Electric utilities and grid operators are interested in integrating EDVs into the electricity grid, instead of treating them as traditional dumb loads, because:

1. Since EDVs run on electricity, a large penetration of EDVs in the market is likely to increase the demand for electricity. Grid operators are concerned that this increased demand might result in an increase in the peak load, which would require them to add additional power generation capacity to the grid [5, 7]. However by using Demand-Side Management commands or pricing signals the grid operators might be able to delay, reduce or switch-off the rate of charging when the demand for electricity is high and push the EDVs to charge during non-peak hours [18]. This would not only reduce the need for new investments but also result in better utilization of the existing power grid.
2. Electric utilities are increasingly diversifying their generation portfolio by adding large quantities of renewable energy resources in order to mitigate climate change and to reduce our dependence on fossil fuels. These resources of electricity, like wind and solar power, are “intermittent” in that the instantaneous power output of these resources depends on the environmental conditions, such as wind speed, at any given time. To match the instantaneous power output of these resources with the instantaneous power demand, the electricity grid needs some form of storage capacity. However, our current electricity grid has a negligible amount of storage capacity, primarily associated with hydro-electric facilities.

Since most vehicles are parked over 90% of the time, these EDVs can be used as a large distributed battery and can provide power storage and ancillary services to the electricity grid when they are not being driven. This concept is known as Vehicle-To-Grid power or V2G power [11, 12]. Although individual EDVs control too little power to sell in the market at an individual level, a large group

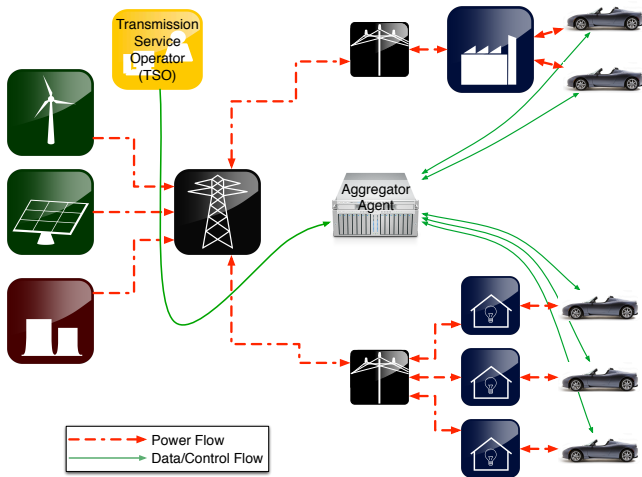


Figure 1: A simplified schematic of the power flow and data flow amongst the entities and agents in our implemented system

of EDVs may form an aggregate or coalition that controls enough power to meaningfully sell, at a profit, in the various electricity markets (See Section 2).

In exchange for integrating with the power grid, the EDV owners would be paid for the power services that they provide and this money can be used to offset or partially subsidize the high cost of the batteries in these EDVs. This in-turn might help to accelerate the adoption of EDVs by price conscious consumers.

In this paper we describe an implemented and deployed multi-agent system that is used to integrate EDVs into the electricity grid managed by PJM. PJM is a Transmission System Operator (TSO)¹ that is responsible for (a) maintaining the security, integrity and reliability of the power grid; and (b) operating wholesale energy markets that enable the transfer of power between the market buyers (consumers) and market sellers (providers). PJM is the largest TSO in the world, servicing 13 states and 51 million customers in the northeastern and midwest United States.

We decided to use a multiagent system because

1. In this system we are dealing with different individuals, systems and entities all of which are self-interested and often have conflicting goals. For example, the TSOs are primarily concerned with the stability and integrity of the grid, while the drivers of the cars are more interested in ensuring that there is sufficient charge in their batteries for whatever trips they are planning to take. These two goals might conflict with each other.
2. Various research topics in multiagent systems, like coalitions, auctions and electronic markets are directly applicable to this problem and offer a way of modeling the EDVs and the power markets that they can operate in.

Our implemented system is able to successfully integrate a group of EDVs into the power grid and is able to provide

¹Alternatively known as an Independent System Operator (ISO) or a Regional Transmission Organization (RTO). We use the term TSO throughout this paper but other papers might use ISO or RTO to refer to the same entity.

both demand-side management and V2G services. We have tested the first phase of the system with 5 EDVs in the PJM TSO providing services in the regulation market. In the next phase, we are using what we have learned to integrate an additional 20 EDVs with plans for an additional 50 EDVs in Phase III.

The outline of the rest of this paper is as follows. In Section 2, we describe the different types of power markets and their suitability for EDVs. Then in Section 3 we describe the agents that form our multiagent system and discuss implementation details. Section 4 describes how coalition formation for EDVs is different from the existing work on the topic and Sections 5 and 6 discuss our evaluation, conclusion and future work.

2. POWER MARKETS

There are a large number of different power markets run by the TSOs, each with a different set of rules and minimum requirements for participation. But generally, power markets can be classified into four distinct types based on the kind of power provided:

1. **Baseload Power:** The baseload power market is for the power that must be provided round-the-clock, usually at low costs per kWh. This kind of power is usually provided using large nuclear, coal-fired, hydroelectric and natural-gas power plants. EDVs are unsuitable for providing baseload power because (a) EDVs have very limited battery and power capacities; and (b) most EDVs are net consumers of power, i.e. they don't actually produce electrical power — they simply store power in the batteries.
2. **Peak Power:** Peak power is generated and purchased at times of exceptionally high demand, usually on hot summer afternoons. Peak power is typically provided by gas generators that can be switched on and off for shorter periods of time, usually 3–5 hours. Whereas EDVs with V2G capabilities might be able to provide peak power, the battery capacities might limit the amount of power that can be economically provided. See [10] for more details.
3. **Spinning Reserves:** Spinning reserves refers to generators that are available to serve the load in case of unplanned events like generator or transmission line failures. They are called spinning reserves because the generators are kept “spinning” and synchronized to the grid so that they are readily available when needed. Since spinning reserves are designed for contingencies, they are rarely used. They might be used 10–20 times a year and even then for durations ranging from 10 minutes to an hour. Furthermore, spinning reserves are paid for the duration they are available even if they are never used.
EDVs with V2G capabilities are highly suited for providing spinning reserves because (a) EDVs can react quickly to contingencies when needed — they can provide power within a couple of seconds when requested and do not need to be kept “spinning” like traditional generators. They just need to be parked and plugged into the grid; and (b) since spinning reserves are rarely called into operation, it does not affect the battery lifetime as much participating in some of the other markets might do.
4. **Regulation:** Regulation power is used to regulate frequency and voltage on the grid by matching the instant-

neous power supplied by the grid with the instantaneous power demand. To provide regulation services and to participate in the regulation market, the participants (typically generators, but in our case, EDVs) must respond to a frequent real-time AGC (Automatic Generation Control) signal sent by the TSO every 2–4 seconds.

There are two types of AGC or regulation signals — (a) *Regulation-Up* signals are sent whenever the demand for power exceeds the supply and are used either to request additional power from the generators/EDVs (i.e. increase the supply) or to switch off some load (i.e. reduce the demand); and (b) *Regulation-Down* signals are sent whenever the supply for power exceeds the demand and are used either to decrease the output of the generators or to increase the load.

To participate in the regulation market, the regulation service providers must first advertise a regulation capacity. This advertisement usually takes the form of a bid in an hourly auction. If the bid is accepted, the regulation providers must respond to the request for specific amounts of power from the TSO.

Some TSOs have separate markets for regulation-up and regulation-down and generators may bid in either one or both of these markets at the same time since the two will never be requested simultaneously.

EDVs are particularly suited to provide regulation power because (a) the batteries in the EDVs can respond very quickly to changes in the regulation request — much faster than traditional generators; and (b) EDVs can participate in the regulation market even if back-feeding of power (i.e. discharging the batteries and providing power back to the grid) is disallowed, by varying the rate at which the batteries are being charged (i.e. by varying the demand.)

For our system, we decided to focus on the regulation market because (a) regulation services command the highest value in the market when compared to other ancillary (for example, spinning reserves) and non-ancillary (for example, peak power) services; (b) EDVs are particularly suited for providing regulation services as discussed above; and (c) the size of the regulation market is larger than the size of the spinning reserve market, so even a large number of EDVs are unlikely to saturate the market.

We worked closely with PJM Interconnection to allow our EDVs to provide regulation services in the PJM TSO. PJM requires a 1MW minimum capacity to bid in the regulation market and requires symmetric advertisements. See Section 3.2 for more details about fulfilling these market requirements. For more information about the PJM market rules, see [3, 19].

3. IMPLEMENTATION

Our agents have been implemented using the JADE (Java Agent DEvelopment Framework) open-source framework [2]. Our system consists of the following agents:

3.1 VSL Agents

The VSL agents run on an embedded linux computer, called the Vehicle Smart Link (VSL), inside the cars. *The VSL agents look after the best interests of the owner or driver of the car.* Since the primary purpose of an EDV

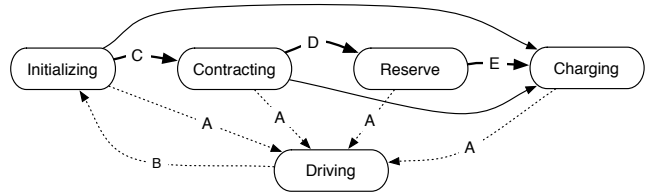


Figure 2: A simplified finite state machine for the VSL agent. The dotted arrow are actions performed by the driver of the EDV and are outside the control of the VSL agent. The dark arrows show the usual progression of state transitions.

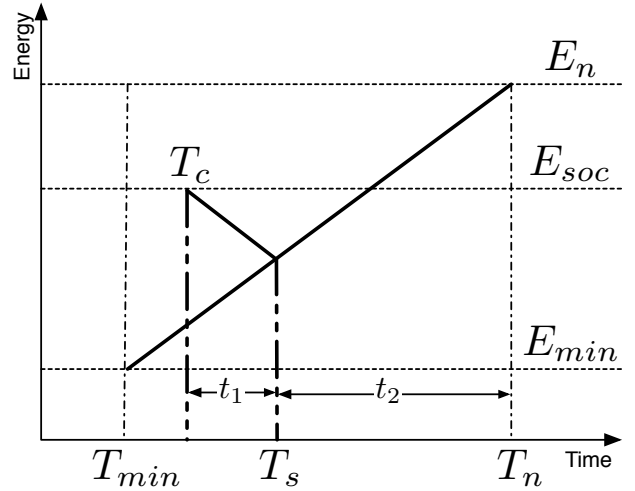


Figure 3: Computing T_s , the time to switch to V2G-monitoring mode

is to be driven, the primary goal of a VSL agent is to ensure that there will always be sufficient charge in the EDVs for whatever trips the driver might want to take. The secondary goal of the VSL agent is to integrate the EDV into the power grid, sell grid services and make money for the driver. We will look at both of these goals below.

The operation of this VSL agent is based on the simplified finite state machine (FSM) shown in Figure 2. In this FSM, the dark arrows show the usual progression of state transitions.

The VSL agent starts out in the *Initializing* state — in this state the VSL agent tries to discover coalition servers servicing the geographic location where the EDV is plugged in. It also predicts the next trip, either by looking it up in the owner’s calendar or by predicting it based on the past driving behavior².

The VSL agent then computes various times for switching between the *Contracting*, *Reserve* and *Charging* states, shown by the transitions *C*, *D* and *E* in Figure 2. These times are computed based on the graph shown in Figure 3. In this figure:

²The prediction algorithms are outside the scope of this paper

E_{soc} = current amount of charge in the battery
 E_n = expected charge required for the **next** trip
 E_{min} = minimum reserved charge
 T_c = current time
 T_n = scheduled time for **next** trip
 T_{min} = time needed to charge from minimum charge
 T_s = time to switch to straight charge mode
 t_1 = time difference between T_c and T_s
 t_2 = time difference between T_s and T_n
 P_{max} = maximum rate of charging
 f_c = charge factor, to account for transmission losses

Based on the above, T_{min} is the time at which charging **must** start if the battery is at its minimum charge, E_{min} , and if we are charging at the maximum possible rate, P_{max} . If $T_c < T_{min}$, the VSL agent can safely participate in V2G regulation without worrying about the next trip since there will always be enough time to charge for the next trip. Hence, whenever $T_c < T_{min}$, the VSL agent is in the *Contracting* state and making contracts with the coalition server.

If $T_c > T_{min}$, the VSL agent switches to the *Reserve* state and will still participate in V2G regulation. However, the VSL agent won't make any contracts and may switch to the *Charging* state at any time depending on the current time, T_c , and the current state of charge, E_{soc} . To determine the time at which the VSL agent should switch to *Charge* state, T_s , the VSL continuously monitors the current state of charge and computes either t_1 or t_2 as shown in Equations 1 and 2.

$$t_1 = \frac{f_c(T_n - T_c)}{(1 + f_c)} - \frac{(E_n - E_{soc})}{[P_{max}(1 + f_c)]} \quad (1)$$

$$t_2 = \frac{(T_n - T_c)}{(1 + f_c)} + \frac{(E_n - E_{soc})}{[P_{max}(1 + f_c)]} \quad (2)$$

In order to sell grid services, the VSL agent must join a coalition with other VSL agents, make contracts with an aggregator agent and dispatch regulation power based on requests received from the aggregator agent. The coalition formation process is described in Section 3.2.1 and the advertisement and dispatch algorithms are described in Section 3.2.2

3.2 Aggregator Agents

The aggregator agent (also known as a coalition server) is responsible for aggregating a group of EDVs, for abstracting away the details about the individual vehicles and for presenting them as a single resource to the TSOs. The aggregator agent is needed because:

1. Individual EDVs command too little power to sell in the power markets and TSOs have minimum requirements for participation in the power markets that they run. For example, the PJM TSO requires a service provider/generator to have a minimum power capacity of at least 1MW to bid in the regulation market. A single EDV, even if it is connected to the grid using a high capacity 240V/80A power line, can only provide a maximum of 19.2kW [13]. Furthermore, EDVs can only store power and can not generate power. This means that the

amount of power they can provide to the grid is limited by the total capacity and the amount of charge that can be stored in their batteries. Hence, a group of EDVs need to aggregate and form a coalition if they are to participate in these power markets.

2. Even if the minimum power requirements could be relaxed, TSOs have traditionally been set up to control large power plants that have an output of several hundred megawatts. To shift from controlling 100 MW power plants to 5 kW cars would require an increase in controlled nodes of 5 orders of magnitude and would require a significant upgrade in the existing software and control systems operated by the TSOs. Therefore, most TSOs³ want a smaller operator to manage the EDVs, and to sell them power in 1 MW to 10 MW blocks. An aggregator agent would represent the best interests of this smaller operator.
3. TSOs require the power resources that bid in their markets to be predictable and reliable. A single EDV, on the other hand, is a very dynamic and unpredictable power resource — since a car may be unplugged and driven at any instant. An aggregator agent is needed to convert a group of dynamic EDVs into a reliable power resource that can bid a predictable amount of power in the power markets.
4. Finally, participating in the PJM regulation market requires an organization to sign a contract with PJM and become a PJM member. The aggregator agent would be operated and controlled by this organization and would be responsible for complying with all the TSO market rules and regulations.

The aggregator agent is responsible for answering questions such as:

- How can a group of vehicles come together to form a coalition?
- How much capacity can a coalition of EDVs report to the grid operators?
- Which vehicles within the coalition should be used to service the power requests?
- How can the money be fairly distributed amongst the coalition participants?

In our deployed system, we have taken steps to address each of these questions, which are described below. However, the problem of integrating EDVs into the electricity grid is a novel problem that opens up new avenues of research within the multiagent community. See Section 4 for more details.

3.2.1 Coalition Formation

To enable coalition formation, the aggregator agent registers with a Directory Facilitator (DF) agent in the JADE framework, which provides the yellow pages service. To join a coalition, the VSL agents first contact the DF agent and request a list of aggregator agents that provide aggregation services for the geographic area in which the EDV is currently located. The DF agent responds with a list of aggregator agents. The VSL agent then contacts each of the aggregator agents in turn and sends a *request-for-contract-terms*

³We have discussed this with PJM, CAISO, NEISO and a TSO in Germany.

message containing, among other information, the identity of the VSL agent, the maximum regulation-up power, the maximum regulation-down power and the battery energy capacity of the EDV. The aggregator agents then verifies the identity of the VSL agent⁴ and then responds with a *contract-terms* message containing an estimate of the expected amount of money the VSL agent might hope to earn during a 24 hour period. The VSL agent then picks the aggregator agent with the best offer and sends it a *join* message.

Once enough VSL agents have joined an aggregator agent to allow it to place a minimum bid in the PJM regulation market, the aggregator agent submits its bids. If a bid is accepted, it dispatches the cars by dividing the received AGC (regulation) signal amongst the VSL agents in the coalition.

Currently we only have 5 EDVs and do not have enough capacity to bid independently in the regulation market. When connected to the grid using a high capacity 240V/80A plug, our EDVs can potentially provide 19.2 kW of power in the regulation market. If the EDVs were permanently connected to the grid, we would need 53 EDVs to meet PJM's 1MW minimum requirement. We have used simulations to empirically determine that we need 230–300 EDVs before an aggregator can reliably provide 1MW of power [9]. To allow us to still participate in the PJM regulation market, we have partnered with a 1MW stationary battery trailer operated by AES Corp. AES Corp. bids 1MW in the regulation market and the received AGS signal is divided proportionally amongst the battery trailer and whatever EDVs are currently connected to the aggregator agent. The aggregator agent is paid according to the Regulation Market Clearing Price (RMCP) for total capacity advertised during a particular hour.

3.2.2 Capacity Advertisement and Vehicle Selection

The amount of capacity that can be advertised by an aggregator agent depends on the makeup of the EDVs that have joined the coalition. Specifically this depends on (a) the plug size used to connect the EDV to the grid; (b) the size of the EDV's battery; and (c) the predictability of the EDV (or the probability the EDV will actually be parked and plugged in during the hour in which a bid is placed.)

Hence, the job of the aggregator agent is to map a population of VSL agents ($P = \{a_1, a_2, \dots, a_n\}$) defined by the tuple $a_i = \langle l, P_{max}, P_{min}, E_{cap}, E_{soc}, E_{min}, \rho \rangle$ to a coalition defined by the tuple $C = \langle A, R_{up}, R_{down} \rangle$, where:

- l is a label identifying a VSL agent
- P_{max} is the maximum rate of *discharge*, in kW, permitted by the plug size and the specific EDV model. We define this quantity as the maximum rate of discharge in order to comply with the generator conventions that denote positive power as that flowing from the generator/EDV to the grid and negative power as flowing from the grid to the EDV batteries. If this value is 0, the EDV is not allowed to back-feed power back to the grid (i.e. the EDV is not permitted to discharge power back to the grid.)

⁴We use the Transport Layer Security (TLS) to allow the VSL agents and the aggregator agents to mutually authenticate each other. Hence, for security and reliability reasons, we only allow communication between known and trusted aggregators and VSL agents.

- P_{min} is the minimum rate of *discharge* permitted, in kW. The quantity will almost always be negative since EDVs are primarily meant to be *charged* from the grid.
- E_{cap} is the battery capacity of the EDV, in kWh.
- E_{soc} is the current state of charge (SOC) in the battery, in kWh.
- E_{min} is the minimum amount of charge needed to maintain a minimum driving range in the EDV and allow the driver to take unscheduled trips.
- $\rho \in R$ is a real number between 0 and 1 denoting the aggregator agent's measure of the predictability of this car.
- $A \subseteq P$ is the set of agents selected for participating in the coalition.
- R_{up} is the amount of regulation up power being advertised by the coalition.
- R_{down} is the amount of regulation down power being advertised by the coalition.

Since the PJM TSO requires symmetric advertisements/contracts, R_{up} must always equal R_{down} . To allow for symmetric contracts we first define a term, Preferred Operating Point (POP) for each individual EDV. To get an intuitive sense for the POP, think of the POP as the *rate of discharge when the regulation request is zero*. (Again we define POP as the rate of *discharge* to comply with generator conventions in which positive quantities indicate power flow from the EDVs to the grid.) For example, if the POP is defined to be -2kW, in the absence of a regulation request, the EDV would be *charging* by drawing 2kW of power from the grid. Now if the TSO sends a regulation request for -3kW (i.e. regulation-down or drawing of power from the grid), the EDV would set its charge rate to -5kW ($-3 + (-2)$), i.e. it would draw 5 kW of power from the grid to charge its batteries.) If instead the TSO sets the regulation request to 1 kW (i.e. regulation-up or providing power to the grid), the EDV would set its charge rate to -1kW ($1 + (-2)$), i.e. the EDV would still charge at 1kW and draw power from the grid, despite providing regulation-up.)

We determine the POP by using the *effective* maximum and minimum rates of discharge, where the effective rate is determined by both the plug size and the state of charge in the battery. We define the POP as follows: (Note in the following description, we assume that δt is the amount of time for which a coalition is required to provide the advertised regulation request.)

$$POP = \frac{E(P_{max}) + E(P_{min})}{2} \quad (3)$$

$$E(P_{max}) = \min \left(\frac{E_{soc} - E_{min}}{\delta t}, P_{max} \right) \quad (4)$$

$$E(P_{min}) = \max \left(\frac{E_{soc} - E_{cap}}{\delta t}, P_{min} \right) \quad (5)$$

Note in Equation 4, if the SOC is at a minimum (i.e. $E_{soc} = E_{min}$), the effective maximum rate of discharge, $E(P_{max})$, will be set to zero and the EDV will only be charged. To see how $E(P_{max})$, $E(P_{min})$ and the POP varies with the SOC, $E(P_{soc})$, see Figure 4. Now to compute the amount regulation-up and regulation-down power that can be advertised by an individual EDV, we define

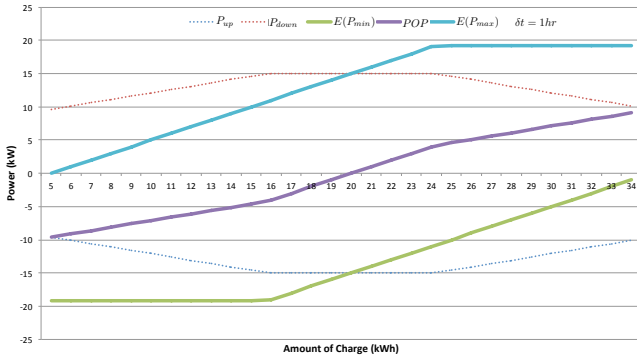


Figure 4: Graph showing the change in Preferred Operating Point (POP) and Advertised Regulation Capacities, P_{up} and P_{down} with the amount of charge in the EDV battery. Note that $E_{cap} = 35kWh$, $E_{min} = 5kWh$ and $\delta t = 1hr$.

$$P_{up} = E(P_{max}) - POP \quad (6)$$

$$P_{down} = E(P_{min}) - POP \quad (7)$$

Again the dotted lines in Figure 4 show how the advertised regulation-up (P_{up}) and regulation-down (P_{down}) power varies with the SOC. Now for the whole coalition, the advertised capacities are calculated as follows:

$$R_{up} = \sum_{a_i \in A} \rho(a_i) \times P_{up}(a_i) \quad (8)$$

$$R_{down} = \sum_{a_i \in A} \rho(a_i) \times P_{down}(a_i) \quad (9)$$

where $\rho(a_i)$ is the availability of agent a_i . Defining the advertised capacity in this way ensures that R_{up} is always equal to R_{down} and allows us to consume all the available capacity while bidding in symmetric markets.

Two questions remain, how do we select the subset of agents to use for the coalition and how do we measure $\rho(a_i)$. Currently, we use all the available agents to form the coalition (that is, we always form a grand coalition). To determine $\rho(a_i)$, we maintain a discounted history of the availability of each EDV, a_i for every hour of the day. Intuitively, we want $\rho(a_i)$ to equal to the probability that a car will be available for a given hour, when it has contracted with the aggregator agent for that particular hour. After every contract hour, we update the value of $\rho(a_i)$ by using the formula

$$\rho(a_i)^{j+i} = \alpha \rho(a_i)^j + (1 - \alpha) \frac{\text{car's availability in min}}{60} \quad (10)$$

3.2.3 Fair-Payoff Division

The traditional solution concept for calculating a fair payoff for a coalition game is the Shapley Value [17]. Given the characteristic function $v(C)$ and the coalition C , the Shapley Value of agent a_j can be calculated by Eq. 11:

$$\phi_j(C, v) = \frac{1}{|C|!} \sum_{S \subseteq C \setminus \{a_j\}} |S|! (|C| - |S| - 1)! [v(S \cup \{a_j\}) - v(S)] \quad (11)$$

As is typical in a large realistic system, calculating the Shapley Value for each agent in this way for any feasible coalition (i.e. a coalition with R_{up} and $R_{down} \geq 1MW$) is clearly intractable, since even the minimal sized such coalition would have 53 EDVs (at 19.2 kW power each) and require computing over 2^{53} subsets of C .

In practice, due to the symmetry axiom of Shapley Values, we might not need to consider every agent as unique and can divide the agents into a set of equivalence classes. However, the number of equivalence classes might still be very large since the number of equivalence classes would depend on (a) the model of the EDV and its battery size; (b) the plug size used to connect the EDV to the grid; (c) the amount of charge in the EDV's battery when contacted out; and (d) the predictability of the EDV.

Given our chosen way of advertising capacities and making contracts, the Shapley value payoff vector will not necessarily be in the core. Consider for example a grand coalition of 54 EDVs where the 54th car has a slightly smaller plug size. While no individual car may have veto power (i.e. any 53-car subset will gain a positive payoff), the 53 cars with the larger plug size would be better off forming their own coalition. Instead, the core is non-empty and contains at least the payoff vector where each car gets a payoff proportional to its own contribution, provided that there are no veto players (cars that can scuttle the coalition on their own) and that the coalition is feasible (meets the minimum requirements). The experimental tendency is that the Shapley value grows closer to the simple proportional payoff as the number of vehicles becomes large (diminishing the effect of the minimum power restriction). Hence, we decided to use a proportional payoff for the EDVs in our system (i.e. each EDV is paid a share proportional to its contribution in the coalition.)

3.3 TSO Agents

The TSO agents communicate with the aggregator agents and provide a wrapper around the legacy systems used by the TSOs. The TSO agents have two main functions — (a) allow the aggregator agents to participate in the regulation market; and (b) send AGC power (regulation) requests to the aggregator agents. The TSO agents are responsible for converting between the legacy modbus data protocols used by the Arcom Director⁵ and the FIPA agent communication language used by our agents.

3.4 EVSE Agents

EVSE is an acronym for *Electric Vehicle Supply Equipment* and is a fancy name for an EDV battery charger. The primary goal of the EVSE agent is to look after the best interests of the owner of the recharging station (which may be the same as the vehicle owner but might also be a completely separate entity like a commercial business or a municipality). Since EDVs are designed to plug into a wide range of power sources, including traditional Edison 125V/15A plugs, the EVSE agents are optional to the operation of the GIVs. When present, the EVSE agents communicate with the VSL agents over a special power + network connector (SAE J1772).

In our system, we define two levels of functionality for the EVSE agents:

⁵The Arcom Director is a remote terminal unit used by PJM.

1. At the basic level, the EVSE agent simply communicates with the VSL agent. The information sent by the EVSE can be very comprehensive and includes information about (a) the maximum charge rate; (b) the maximum discharge rate; (c) whether or not V2G (or back-feeding of power to the grid) is permitted at this charger; (d) meter id and transformer id of the circuit on which this EVSE is located⁶; (e) the network settings the VSL agent should use for internet access; (f) whether the VSL should provide emergency power in the case of a power disruption; and (g) error codes related to ground, control and pilot faults.
2. At a more advanced level, the EVSE can negotiate charging contracts with the VSL agents. This would be useful in situations where the EVSE are owned by a third party such as a business or a city. (For example, there are plans for installing EVSEs on our main street and in our interstate service centers.) In these situations, the EVSE agents would negotiate rates of charging and the corresponding costs with the VSL agents and, in some cases, might even allow the EDVs to charge for free as long as the VSL agents agree to forego the revenue earned by providing GIV services. These EVSE agents would also be responsible for keeping records of charging events and for billing customers appropriately for the services used.

At present we have only implemented the basic functionality in our EVSE agents and are working on incorporating the more advanced functionality into future releases.

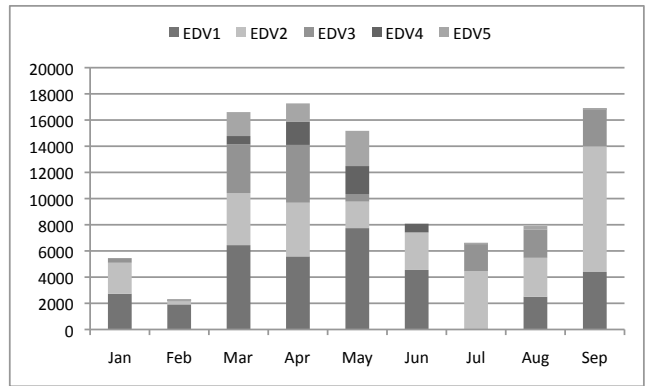
4. RESEARCH CHALLENGES

Coalition formation has been studied extensively in both game theory [8] and multiagent systems (see [16, 6, 8, 14, 1] for a small sampling) and has even been applied to power transmission planning [4] and open environments [15]. We believe there are some key differences between the existing research on coalition formation and the kind of coalition formation that we are interested in in this paper:

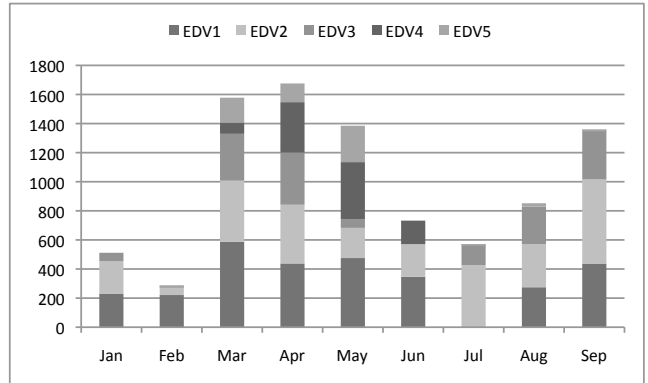
- Coalition formation for EDVs differs from iterative coalition formation games in that each game is not independent. Rather the coalition’s ability to participate in the next game depends on the actions of the previous game. If we use up all the charge in the EDVs in game i , we won’t have any charged EDVs available for use in game $i + 1$. Hence, what we do in one hour affects the kind of coalition that we can form in the next hour. Furthermore, the set of agents that form the coalition is highly unpredictable and varies from hour-to-hour.
- Most coalition games assume a fixed characteristic function that defines the payoffs received by the formed coalitions. However, in the case of coalition formation for EDVs the characteristic function is not fixed. Instead part of the problem is determining the amount of capacity to bid in the different markets which involves determining an appropriate characteristic function for the coalitions.

We believe these two issues need to be modeled and studied by the multiagent community in general and we would like to focus on modeling these issues in our future work.

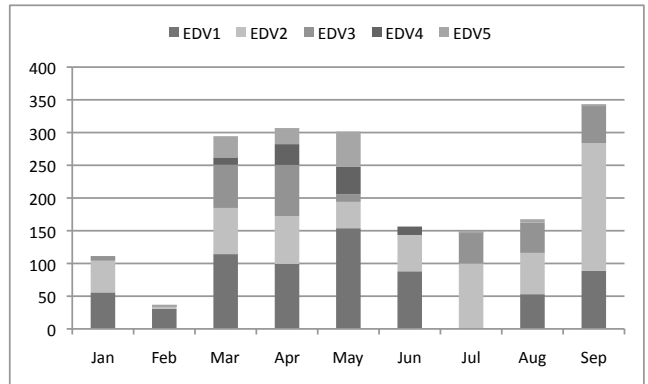
⁶This information together with the topology of the power grid can be used by the aggregator agent to limit the maximum load on an individual transformer.



(a) Regulation Capacity Offered (in kW-h)



(b) Total number of hours plugged in



(c) Amount of money earned by the EDVs (in US\$)

Figure 5: Graphs showing the performance of our five EDVs for the first nine months of 2010.

5. EVALUATION

Since this paper describes an implemented and deployed system, we thought the best way to evaluate our system would be to describe its operation over the first nine months of this year. Figure 5a shows the total capacity bid by our EDVs in kW-h⁷; Figure 5b shows the number of hours that

⁷Since the regulation market pays for the bid regulation capacity and not for the actual power provided, the unit for regulation capacity is kW-h. One kW-h is a unit of power capacity meaning that one kW of regulation power capacity

each EDV was plugged-in and providing regulation services; and Figure 5c shows the amount of money earned by our EDVs, in US dollars, during the same period.

As can be seen, the amount of regulation capacity offered and the amount of money earned is directly proportional to the number of hours plugged in. (These EDVs were mostly plugged into 208V/50A plugs although occasionally an 80A plug was also used.) We started out with 3 EDVs in January and February. Since the winter months (especially February) were particularly severe in the northeast (in 2010), we chose to not participate in the regulation market in order to conserve the battery life of our EDVs. In March, we added two more EDVs to give us a total of 5 EDVs. The amount of regulation offered dropped significantly in the months of June, July and the starting of August because we were making significant upgrades to our system.

Extrapolating from our data, if an EDV is plugged-in and providing regulation services for 15 hours a day, it can expect to make a hundred dollars a month given the current Regulation Market Clearing Prices (RMCPs). Given that the RMCP was twice as high as what it is right now before the start of our current economic recession, EDVs owners can expect to make between 100 and 200 dollars a month or between 1,200 and 2,400 dollars a year by participating in the regulation market. This is a significant amount of money that can be used to offset the high costs of EDVs.

6. CONCLUSION AND FUTURE WORK

This paper describes an implemented and deployed system for integrating a group of EDVs into the electricity grid. We motivated the problem, described the various types of power markets and presented an implementation of a multiagent system that allows EDVs to participate in the regulation market. We have also deployed 5 EDVs in the PJM TSO that, in conjunction with a 1MW battery trailer operated by AES Corp., has been able to bid and earn money in the regulation market.

For our future work:

- We plan to deploy another 50 EDVs within the next two years. Our goal is to have enough EDVs so that we may participate independently in the PJM regulation market. We would also like to study the scalability of our approach to a couple of thousand EDVs.
- We would like to focus on each of the open research challenges presented in Section 4.
- We would like to lead the effort to develop a standard set of protocols for (a) communicating between the VSL agents and the Vehicle Management Systems (VMSs) inside the EDVs; and (b) communicating between the aggregator agents and the VSL agents. This latter would involve defining a standard ontology for this application.

7. ACKNOWLEDGEMENTS

This work was supported by funding from the U.S. Department of Energy, Office of Electricity Delivery and Reliability (grant number DE-FC26-08NT01905, Willett Kempton, PI). We also acknowledge support on this aspect of the V2G project from Pepco Holdings, Inc., PJM Interconnection, AC Propulsion, Inc., and AES Corporation.

was available for one hour.

8. REFERENCES

- [1] Y. Bachrach and J. S. Rosenschein. Coalitional skill games. In *AAMAS '08*, pages 1023–1030.
- [2] F. L. Bellifemine, G. Caire, and D. Greenwood. *Developing Multi-Agent Systems with JADE*. Wiley Series in Agent Technology. Wiley, 2007.
- [3] M. Bryson. *PJM Manual 12: PJM Manual for Balancing Operations*. PJM Intercon., August 2009.
- [4] J. Contreras, M. Klusch, and J. Yen. Multi-agent coalition formation in power transmission planning: a bilateral shapley value approach. In *In Proc. of 2nd Intl. Conference on Practical Applications of Multi-Agent Systems, PAAM*, pages 21–23, 1998.
- [5] N. DeForest et. al., Impact of Widespread Electric Vehicle Adoption on the Electrical Utility Business – Threats and Opportunities. Technical Report 2009.5.v.1.1, Center for Entrepreneurship and Technology, University of California, Berkeley.
- [6] S. S. Fatima, M. Wooldridge, and N. R. Jennings. A linear approximation method for the shapley value. *Artif. Intell.*, 172(14):1673–1699, 2008.
- [7] K. Fell. Assessment of plug-in electric vehicle integration with iso/rto systems. Technical report, ISO/RTO Council, March 2010.
- [8] J. P. Kahan and A. Rapoport. *Theories of coalition formation*. L. Erlbaum Associates, 1984.
- [9] S. Kamboj, et. al. Exploring the formation of Electric Vehicle Coalitions for Vehicle-To-Grid Power Regulation. *AAMAS workshop on Agent Technologies for Energy Systems (ATES 2010)*.
- [10] W. Kempton and T. Kubo. Electric-drive vehicles for peak power in japan. *Energy Policy*, 28(1):9 – 18, 2000.
- [11] W. Kempton and S. Letendre. Electric vehicles as a new source of power for electric utilities. *Transportation Research*, 2(3):157–175, 1997.
- [12] W. Kempton and J. Tomic. Vehicle-to-grid power fundamentals: Calculating capacity and net revenue. *Journal of Power Sources*, 144(1):268 – 279, 2005.
- [13] W. Kempton and J. Tomic. Vehicle-to-grid power implementation: From stabilizing the grid to supporting large-scale renewable energy. *Journal of Power Sources*, 144(1):280 – 294, 2005.
- [14] K. Lerman and O. Shehory. Coalition formation for large-scale electronic markets. In *4th Inter. Conf. on MultiAgent Systems, 2000*, pages 167–174, 2000.
- [15] N. Ohta et. al., Anonymity-proof shapley value: extending shapley value for coalitional games in open environments. In *AAMAS '08*, pages 927–934.
- [16] T. W. Sandholm and V. R. Lesser. Coalitions among computationally bounded agents. *Artif. Intell.*, 94(1-2):99–137, 1997.
- [17] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.
- [18] S. Vandael, N. Boucké, T. Holvoet, and G. Deconinck. Decentralized demand side management of plug-in hybrid vehicles in a Smart Grid. *AAMAS workshop on Agent Technologies for Energy Systems (ATES 2010)*.
- [19] M. Ward. *Manual 11: PJM Manual for Scheduling Operations*. PJM Interconnection, January 2010.

Multi-Agent Monte Carlo Go

Leandro Soriano Marcolino
Matsubara Laboratory – Intelligence Information
Science Department
Future University of Hakodate
Hakodate, Japan
g2209001@fun.ac.jp

Hitoshi Matsubara
Matsubara Laboratory – Intelligence Information
Science Department
Future University of Hakodate
Hakodate, Japan
matsubar@fun.ac.jp

ABSTRACT

In this paper we propose a Multi-Agent version of UCT Monte Carlo Go. We use the emergent behavior of a great number of simple agents to increase the quality of the Monte Carlo simulations, increasing the strength of the artificial player as a whole. Instead of one agent playing against itself, different agents play in the simulation phase of the algorithm, leading to a better exploration of the search space. We could significantly overcome Fuego, a top Computer Go software. Emergent behavior seems to be the next step of Computer Go development.

Categories and Subject Descriptors

I.2.1 [Artificial Intelligence]: Applications and Expert Systems—Games

General Terms

Algorithms, Experimentation

Keywords

Emergent Behaviour, Collective Intelligence

1. INTRODUCTION

Go is a two-player turn-based strategy board game, that is famous for being one of the main challenges in Artificial Intelligence. A small set of simple rules¹ leads to a game amazingly complex for a human being and a search tree that is unbearably large for a computer. There are many reasons for this difficulty of developing a strong artificial player. First, Go is played in a large board, 19x19, with 361 intersections, creating difficulties for tree search based algorithms. Second, generally most of the intersections are valid movements, increasing the number of possible states from a given state of the board. Third, the stones interact in complex ways during the game; one stone may influence a distant group, for example in situations where there is a

¹Available at many places, for example: <http://www.pandanet.co.jp/English>

Cite as: Multi-Agent Monte Carlo Go, Leandro Soriano Marcolino, and Hitoshi Matsubara, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 21-28.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

ladder. Besides, building an evaluation function is not trivial. Even end of game situations, that intuitively should be simpler, were proved to be PSPACE-hard [31]. According to [1], compared to the complexity of Chess (10^{50}), the complexity of Go (10^{160}) is bigger by a factor of 10^{110} . We can see, therefore, how challenging it is to create an artificial player of Go.

However, recently, with the development of evaluations of the board state based on simulations (known as Monte Carlo techniques), the strength of Computer Go players improved significantly. Thanks to artificial players like MoGo, Crazy Stone, Fuego, Many Faces of Go, and Zen, the best Go programs are now considered amateur level 2 dan. Further improvement was achieved by parallelization, as it increases the computational power, allowing a deeper exploration of the possible movements. In February 2009, Many Faces of Go, running on a 32-core Xeon cluster, beat the professional player James Kerwin, in a 19x19 board with a handicap of 7 stones. Many recent works are now investing in the parallelization of Monte Carlo techniques. However, there is always a limit in the amount of speed-up that can be gained in a parallelization design.

Generally, there are two ways to increase the strength of an artificial player: advances in computational power, which can be achieved by parallelization, and advances in the theory, which can be achieved by new algorithms and methods. Nowadays, the research in Monte Carlo techniques seems to be focused on the parallelization of the current approaches. However, it is always desirable to advance the theory with the creation of better algorithms, that lead to stronger players even when the computational power has not necessarily increased. We believe that the next theoretical step lies in the investigation of Multi-Agent methodologies.

Multi-Agent systems have been used to solve a great range of problems in Artificial Intelligence. The emergent behavior of a great number of simple agents have been applied in algorithms like *Ant Colony Optimization* [11], *Particle Swarm Optimization* [20], etc, in order to solve difficult optimization problems. It is also notable how emergence can lead to complex and intricate group behavior [21, 22, 23, 28].

Emergence is a powerful concept, not only in Computer Science, but also in a variety of disciplines, like philosophy, systems theory and art. The stock market and the Internet are important systems to modern life that arise thanks to the emergence of simple components. Emergence is also fundamental in biological systems. A notable example is an ant colony. It is known that the queen does not order directly the ants. Each ant is always reacting to stimuli generated

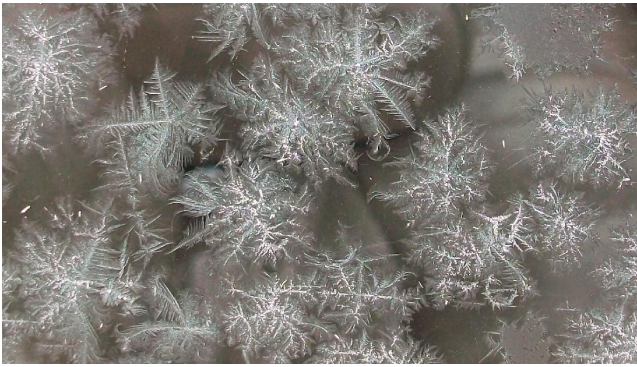


Figure 1: Water crystals, formed by a natural emergent process (taken from www.wikipedia.org).

by chemical scent from larvae, other ants, intruders, food, waste, etc, and they leave chemical that will be used as stimuli to other ants. Therefore, there is no centralized control, but the ant colonies exhibit complex behavior and are able to solve complex problems. Another example is the formation of water crystals on glass, a natural emergent process created by the random motion of water molecules, that leads to a highly-organized structure (Figure 1).

However, emergence is generally not a clear concept. In this paper we define emergence as a great number of simple iterations that occur in a system, leading to a complex result. We can model the Monte Carlo evaluations as one agent that repetitively plays against itself using a playout strategy. Although the playout strategy might be simple, the combination of a great number of games with a tree exploration phase makes intelligent game play emerge in a Monte Carlo Tree Search algorithm (MCTS). In this paper we explore this further, by evaluating the effects of having not only one, but many different agents at the playout phase of a MCTS.

At each stage of the Go board, one agent is selected to generate a movement, leading the board to the next stage. The agents act in turn, therefore there is no spatial organization, but a temporal organization. However, each agent acts in the environment that was left by the previous one, and this interaction seems to lead to a higher playing strength. As the interactions are simple, but they lead to something complex (high-level go), we believe emergence is a good concept to define our idea.

Our proposed algorithm is also inspired by the advantages of diversity. It is currently believed by some social scientists and economists that the best teams are not necessarily composed of the best individuals. In order to build a team that is effective in solving problems, it is also important to look for diversity, to bring together people with different perspectives and solution strategies [27]. By using different agents during the simulation process, we are also exploring this concept, but in a Multi-Agent context.

We modify Fuego [13], an open source implementation of a powerful MCTS algorithm: UCT Monte Carlo Go. Therefore, the contribution of this paper is to offer a new paradigm for the exploration of Monte Carlo Go. Our experimental analysis show that we could significantly overcome Fuego, and produce a stronger Computer Go program.

2. RELATED WORK

A great variety of approaches have been proposed in the literature in order to tackle with the complexity of Go. The problem is too difficult for a conventional α - β search, forcing the researchers to try many different methods. An interesting survey of the literature can be found in [6]. Classical works used abstractions [15] and patterns [4]. Other important approaches that have been explored include learning [8], cognitive modeling [5] or combinatorial game theory [24].

Generally, in the classical way to develop a Go program, specific game knowledge has to be implemented. Therefore, many algorithms were proposed to resolve specific subproblems of the game [3]. However, the Monte Carlo approach appeared, which originally used only the simple Go rules to perform random simulations in order to discover good positions to play [7]. Later, the Monte Carlo simulations were used to evaluate leaves in tree search algorithms, and the simulations started to use heuristics, which included some Go knowledge, in order to improve their realism, as in [12]. The state of the art was further advanced by the UCT Monte Carlo algorithm [17], which contributed with significant improvements in playing strength. The proposed program, MoGo, won all the tournaments on the international Kiseido Go Server² on October and November 2006.

In order to achieve further enhancements, parallel and distributed versions of the game started to appear on the literature. Generally, the idea is to use a great number of machines or processors to increase computation power. According to [10], three different parallelization approaches are possible in UCT Monte Carlo: root parallelization, leaf parallelization and tree parallelization. In root parallelization each thread is responsible for one tree, and when the time is finished, the results are merged. In leaf parallelization, many simulations are executed to evaluate a single leaf, each one by one thread. In tree parallelization, many threads execute in a single, shared tree. In [16] it is proposed a straight algorithm for multi-core parallelization, based on shared memory, and an algorithm for cluster parallelization that uses less messages than a simple generalization of the multi-core algorithm. The multi-core algorithm achieved a 63% percentage of victory by doubling the computational power and the cluster algorithm achieved 83.8% percentage of victory by using 9 machines. Some works propose distributed systems based on a client/server architecture in order to increase the number of available playouts [18]. Recently, a top Computer Go program, Zen, was run in a large cluster of computers [19]. A similar approach is also investigated by [9], where a percentage of victory of 70.50% could be achieved against GnuGo, using 16 slaves. However, the results do not improve with a higher number of slaves, and even decreased in some cases. Root parallelization in the Fuego system was studied by [29], where experiments with 64 cores demonstrated that although the program gets stronger, there are limitations in the possible performance gain.

Recently, distributed versions of the top Computer Go programs have won against professional players in handicap games. However, it is known that the overhead of the parallelization imposes a limit to the possible improvement in game strength. In [18], for example, in 9x9 boards the system saturated with 7 servers, and the use of 4 servers brought a speed-up factor of only 1.55. In [10], tree par-

²<http://www.weddslist.com/kgs/past/index.html>

allelization only scaled well up to 4 threads. A lock-free parallelization was proposed by [14], but it could not scale beyond 7 threads.

The next step seems to be converging into Multi-Agent System paradigms. Some works started to apply this idea, but in order to play other games. In [25], a consultation system to play Shogi is proposed. A set of players send their opinion about what should be the next movement, and one of the opinions is selected as the official movement. The authors show that a consultation system composed of three famous Shogi programs plays better than each software individually. In [30] the authors extend the last approach, but this time they use the position evaluation of different players in order to select a single movement. The number of agents in these works was limited, though, with at most 6 agents. In [26], the authors explore a Swarm Intelligence Algorithm, Stochastic Diffusion Search, to build an artificial Othello player. We believe that the use of Multi-Agent Systems has to be further explored, and it can be the next cornerstone in Computer Go development.

Some social scientists and economists currently believe that teams of diverse people can have strong characteristics for solving difficult problems [27]. By combining different perspectives and solution strategies, a diverse team can explore a greater range of possible solutions for a problem; while a team with high-talented but similar individuals might not be able to explore so many different solutions, as each member will tend to have similar results as the other members of the group. Therefore, a team of diverse members might perform better than a team with the best individuals. This concept is also an important point to be explored in the development of Multi-Agent paradigms for Computer Go.

In this work we are going to extend the top MCTS algorithm, UCT Monte Carlo Go, with a Multi-Agent System paradigm. Instead of showing the computational power gains that can be obtained by parallelization or distribution, we are going to show how the emergent properties of a great number of simple (and diverse) agents, by itself, can enhance the strength of an artificial Go player.

3. METHODOLOGY

First, we are going to introduce UCT Monte Carlo Go. The algorithm is based on the multi-arm bandit problem. A multi-arm bandit is like a traditional slot machine, but with many arms. Each arm has a reward drawn from an unknown probability distribution. The objective is to maximize the total sum of iterative plays. When choosing an arm to play, there is a balance between selecting the best arm found so far, or exploring other arms. In [2], it is proposed a simple algorithm, called UCB1 in order to solve the selection problem. Let's define the K-armed bandit problem by the random variables $X_{i,n}$, for $1 \leq i \leq K$ and $n \geq 1$. Each variable is the reward of arm i when it is played at time n . Given a certain arm i , the rewards $X_{i,n}$ are independent for all n , and are identically distributed according to an unknown probability distribution. The rewards across arms are also independent, but they might not be identically distributed.

The algorithm selects the arm j , that maximizes $\bar{X}_j + \sqrt{\frac{2 \log n}{T_j(n)}}$, where n is the overall number of plays up to the current iteration, $T_j(n)$ is the number of times arm j has been played after the first n plays, and \bar{X}_j is the mean of

the values obtained so far when arm j was selected. In [2], it is also introduced a slightly more complicated algorithm, called UCB1-TUNED, that had better experimental results. First, they calculate an estimation of the upper bound on the variance of arm j , by:

$$V_j = \left(\frac{1}{T_j(n)} \sum_{y=1}^{T_j(n)} X_{j,y}^2 \right) - \bar{X}_j^2 + \sqrt{\frac{2 \log n}{T_j(n)}}$$

Then, they select the arm j that maximizes the following equation:

$$\bar{X}_j + \sqrt{\frac{\log n}{T_j(n)} \min\{1/4, V_j\}} \quad (1)$$

In UCT Monte Carlo Go, each Go board situation is seen as a bandit, and each possible move is seen as an arm with unknown reward of a certain distribution. Generally, the algorithm can be defined by two phases: tree search and leaf evaluation (also known as playout). The tree search phase starts at the root of the tree. At each node (Go board situation), the child-node (possible move) that maximizes Equation 1 (UCB1-TUNED) is selected as the next node to be visited. This is executed recursively, always choosing the child-node according to UCB1-TUNED. When a node is selected that has never been visited before, the next phase is executed: score estimation by Monte Carlo simulations, where heuristic-driven random games are executed from the state of the leaf until the end of the game. Generally the heuristics are designed in a way that the end game can be easily recognized, and the final score easily calculated. The final score is used to estimate the value of the leaf. The value of the nodes in the path are then updated iteratively, from the father-node of the selected leaf to the root. Note that the Go board states created during the Monte Carlo simulations will not become part of the tree, they are used only to estimate the value of the leaf. Improving the quality of the simulations will improve the estimation of the score, leading to a stronger player [32].

We can model the random simulations as one agent playing against itself using its available heuristics (Figure 2(a)). In this work, we investigate the effects of having not only one, but several agents playing against each other (Figure 2(b)). Each agent has a different playing style, increasing the range of exploration of the search space. As will be further explained, at every stage of the simulation process, a different agent will be selected in an *agent database*, and this agent will be responsible for selecting the next movement. Note, therefore, that (contrary to our first idea) in our approach we are not executing a tournament between different agents, as one agent does not play a full game against another.

We based our implementation on Fuego, an open source UCT Monte Carlo Go algorithm. The Fuego system executes several heuristics hierarchically. It starts by selecting the first heuristic. In case it cannot generate a movement, it proceeds by selecting the next one on the hierarchy. The process repeats until a heuristic generates a movement. If no heuristic can generate a movement, a random move is selected from the board. Generally the heuristics are applied in the neighborhood of the last movement. The current version of Fuego (0.4) has mainly five heuristics: **Nakade** If there is a region of three empty points, generates a move-

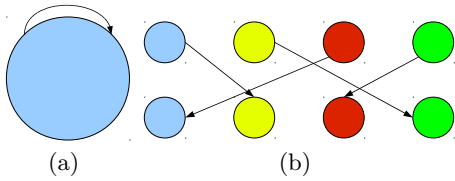


Figure 2: Original single-agent Monte Carlo (a) and proposed Multi-Agent Monte Carlo (b). The colors represent different agents, and the arrows represent interaction.

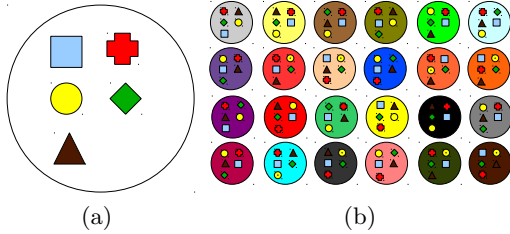


Figure 3: Original Fuego agent (a) and new *agent database* (b).

ment in the center of this region; **Atari Capture** Captures an Atari; **Atari Defend** Defends an Atari; **Lowlib** Move generator for 2-liberty blocks; **Pattern** Uses a set of 3x3 patterns, this heuristic is applied in the neighborhood of the two last moves.

The hierarchical order of the heuristics is fixed. A representation of the Fuego original agent can be seen in Figure 3(a), where each symbol represents a different heuristic, and the order of the symbols represent the order that each heuristic will be applied. We created several new agents in the Fuego system by changing the order of the default heuristics of the original agent. Therefore, each agent will give a different priority to the heuristics; which will make each agent have a different playing style (Figure 3(b)). The set of all agents implemented in the system form an *agent database*.

Every time one movement will be generated during the Monte Carlo simulations, one agent is randomly selected in the *agent database* and this agent will be responsible for selecting the movement. Therefore, at each step in the simulation process, a different agent is going to decide the next movement on the board (Figure 4). This approach allows the Monte Carlo method to explore better the search space, using the same amount of computation time. The intuition behind this idea is simple. Although some Go movements, such as the capture of a stone, can seem to be quite strong for a beginner, an experienced player knows that preferring apparently “strong” movements all the time will lead to a poor and unnatural game. Therefore, in order to simulate more realistic Go games, it is necessary to diversify the movement generation process.

However, although we can use 120 different agents, we empirically found out that using all of them does not lead to a stronger player (see Section 4). It is necessary to select a set of agents that effectively lead to better playing abilities. As testing all possible combinations of agents is very expensive, we executed a simple greedy learning algorithm. We start

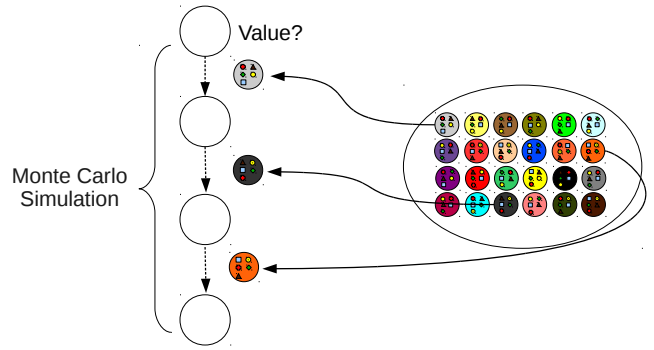


Figure 4: Agent selection in the Monte Carlo simulation process.

with only the original Fuego agent in the database. Then, we perform a series of games against Fuego. The result (percentage of victory) is saved. We then add one more agent in the database. A series of games is again performed against Fuego. If the result is better than the best result found so far, the agent will remain in the database. If the result gets worse, the agent will be removed from the database, and will not be tested anymore. The algorithm proceeds by testing all the remaining possible agents. Note that every time a “good agent” is found, it will be permanently inserted in the *agent database*, and it will be used in all the following iterations of the learning process. Also note that the original Fuego agent will always be in the *agent database*, because it is used in the first iteration.

Therefore, our algorithm is a hill climbing in the space of agent sets: we add one agent to the set and greedily keep it if the new set performs better. We test each agent exactly one time. The most natural way is to generate a random list in which all agents appear exactly once, and follow the order of the list. However, we also manually changed the list in one of our experiments, in order to try to achieve a better solution. As will be seen in the next section, our simple learning algorithm led to a significant percentage of victory against Fuego, showing that Multi-Agent Monte Carlo Go can effectively be used to create stronger players.

4. RESULTS

In this section we are going to present the experiments performed to validate our approach. They were all executed in a 9x9 board, with the same time limit for both our system and Fuego. We used Fuego’s default time limit and default configuration for the number of playouts per leaf (1 playout per leaf, for a 9x9 board). We executed 500 games with our system playing as White, and 500 games with our system playing as Black, giving a total of 1000 games per configuration. The experiments were executed in a cluster of Intel(R) Xeon(R) CPU E5530, at 2.4GHz and with 24GB of RAM. Note that our algorithm is not parallel, but we used a cluster in order to distribute the execution of the 1000 games, decreasing significantly the time necessary to run the experiments. The cluster used is part of the InTrigger³ platform, a cluster of more than 13 clusters distributed across Japan. They are intended to be used for information technology research, both for system software and for large scale data

³<http://www.intrigger.jp>

N	AC	AD	L	P
AD	N	AC	P	L
AD	N	P	AC	L
AD	AC	P	N	L
N	AC	P	L	AD

Table 1: Selected *agent database*.

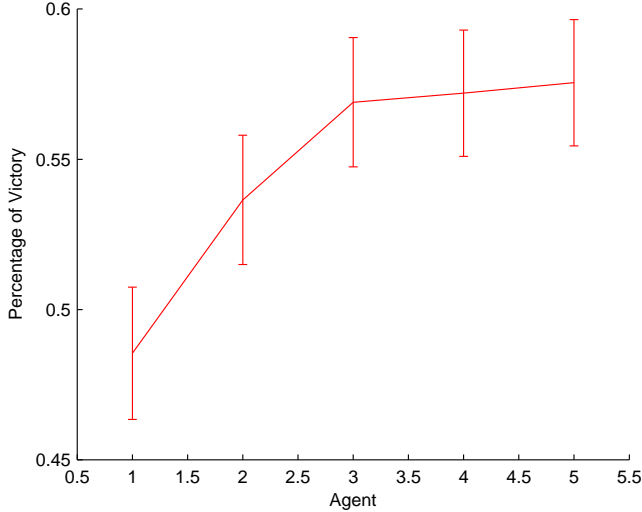


Figure 5: Percentage of victory for the selected *agent database*.

processing researchers.

We first ran our algorithm with all the possible 120 agents. It led to a relatively low percentage of victory: 41.20% ($\pm 2.10\%$). After performing several experiments with the database, we found out that some agents seemed to decrease, while other agents seemed to increase the percentage of victory. Therefore, we created a simple learning algorithm, that tries to add each agent in the database, and tests if it increases or decreases the strength, as described in the previous section.

First, we are going to show our results when the order in which each agent is tested is random. Let N , AC , AD , L and P be the Nakade, Atari Capture, Atari Defense, Lowlib and Pattern heuristics, respectively. The *agent database* selected by the learning algorithm is represented in Table 1, where each line defines one agent and the columns defines the order in which each heuristic is attempted. The first line corresponds to the original Fuego agent. Our algorithm was able to find a set of 5 agents that seems to increase playing strength.

The result obtained with the addition of each new agent can be seen in Figure 5. As can be observed, from a 48.55% ($\pm 2.20\%$) percentage of victory with only Fuego’s original agent, with 5 agents we could achieve 57.55% ($\pm 2.10\%$), a gain of 9.00%. Therefore, our strategy seems to be effective into improving the strength of Computer Go algorithms. We performed a t -test analysis that showed that the result with 5 agents is better than the result with only Fuego’s original agent with 99% of confidence.

The result of about 48% when our system has only the Fuego original agent is a little bit different from the theoretically expected 50%. We believe this might happen be-

Agent Number	Percentage of Victory
0	48.50% \pm 2.20%
5 (α)	52.85% \pm 2.15%
6 (β)	53.60% \pm 2.15%
64	57.30% \pm 2.15%
70	29.60% \pm 1.90%

Table 2: Percentage of victory for each individual agent.

cause the game with only one agent is not really “Fuego” vs. “Fuego”, it is “Fuego” vs. “Fuego with a small overhead”, as the algorithm for agent selection and agent execution is still there, and it is ran in every step of the Monte Carlo simulations. As the number of simulations is very high, we believe this overhead might be responsible for the 48.55% result, instead of 50%.

We also executed games with our system running with a single agent (again, against Fuego). In each execution, we used one of the agents that were selected for the *agent database*, but only that one. The objective of these experiments is to see if the result of the agents as a group is better than the result of each individual agent. We can see the percentage of victory obtained for each agent in Table 2, where the Agent Number represents the position of the agent in the list (or, in other words, the number of the iteration in which the agent was tested). We called agent 5 as α and agent 6 as β because they are going to appear again in our next experiment.

Many interesting observations can be drawn from these experiments. First, as can be seen, the result of the group (57.55%) was better than the result of each individual agent, though the difference between the group and the agent 64 is quite small. However, even before adding agent 64, the group already performed quite well (56.90%), a percentage of victory higher than each member. Second, agent 70 is clearly much weaker than the other agents, but when it was added in the *agent database* the result improved 0.35%, instead of decreasing. Therefore, it seems that there is a group phenomena that makes the algorithm stronger.

The learning graph of our algorithm can be seen in Figure 6. After adding agent 5 and 6, the system fluctuates, and is able to escape from the local minimum (lack of improvement) only with the addition of agent 64. After adding agent 70, the system fluctuates again and is not able to find a better solution.

We ran our algorithm a second time, but now we tested the agents in a different order. Before we developed our learning algorithm, we had a list of 15 agents that we believed to be strong (by intuition and trial an error experiments), and we moved those agents to the beginning of the list. Our original intention, when we developed the learning algorithm, was to test this set of agents. The rest of the agents followed the same order as the previous experiment. The agents that compose the new solution found by the learning algorithm can be seen in Table 3. The result obtained with the addition of each new agent is represented in Figure 7, and the learning graph can be seen in Figure 8. This time, we found a slightly better result, of 59.15% ($\pm 2.10\%$).

We executed games with our system running with a single agent. The percentage of victory obtained for each agent can be seen in Table 4. The Agent Number of each agent is

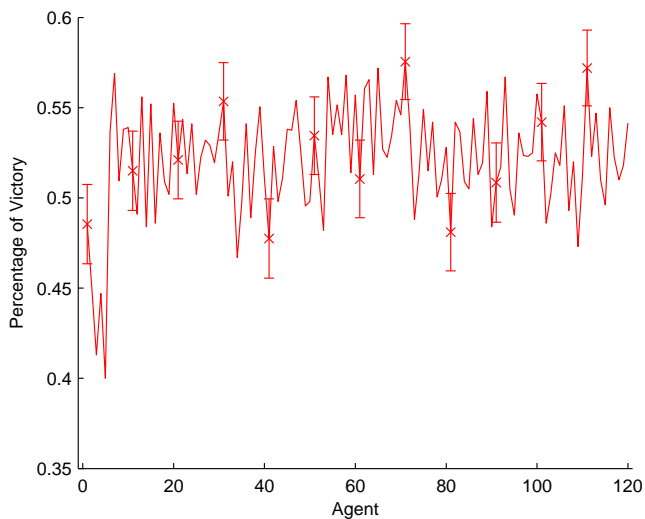


Figure 6: Learning graph, as the algorithm tries to add each agent in the database.

N	AC	AD	L	P
AD	N	AC	P	L
AD	N	P	AC	L
AD	P	L	N	AC
AC	N	AD	L	P

Table 3: Selected *agent database*, in the not random order.

different than last time, as the order changed, but agent 1 and 2 are the same as agent 5 and 6 of the last experiment, respectively. Therefore, we named them α and β . Again, the result of the group was better than the result of each individual agent (although the difference between the group and agent 3 is small). This time, the difference between the group and the best agent seems to be higher than in the previous experiment. And, for the second time, agents that are weaker were able to increase the percentage of victory when they were added to the group. Agent 42 had a percentage of victory of only 50.90%, but was able to increase the percentage of victory of the system in 1.20% when it was added in the group. Therefore, with this new agent order, we were also able to show that we can increase the strength of Monte Carlo Go using the emergent behavior of a group of agents, this time with a slightly better result.

As can be seen, we could find two agent sets that perform quite well against the original Fuego. After analyzing the result of our experiments, we think we have strong indications that the emergent behavior of a group of agents can

Agent Number	Percentage of Victory
0	48.55% \pm 2.20%
1 (α)	54.40% \pm 2.15%
2 (β)	54.55% \pm 2.15%
3	57.05% \pm 2.15%
42	50.90% \pm 2.20%

Table 4: Percentage of victory for each individual agent, in the not random order.

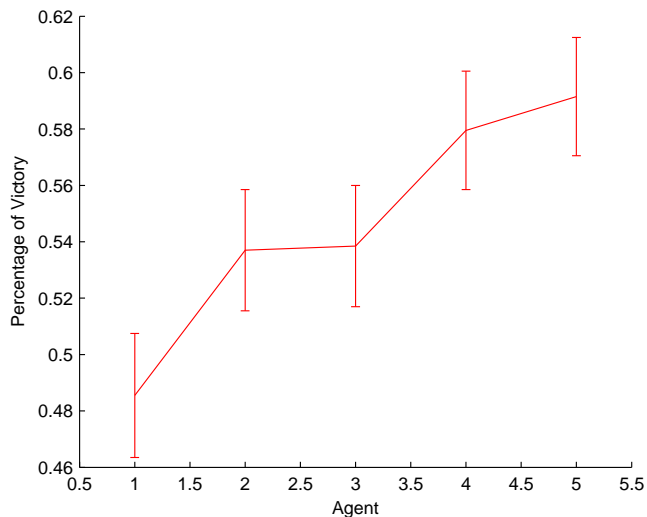


Figure 7: Percentage of victory for the selected *agent database*, in the not random order.

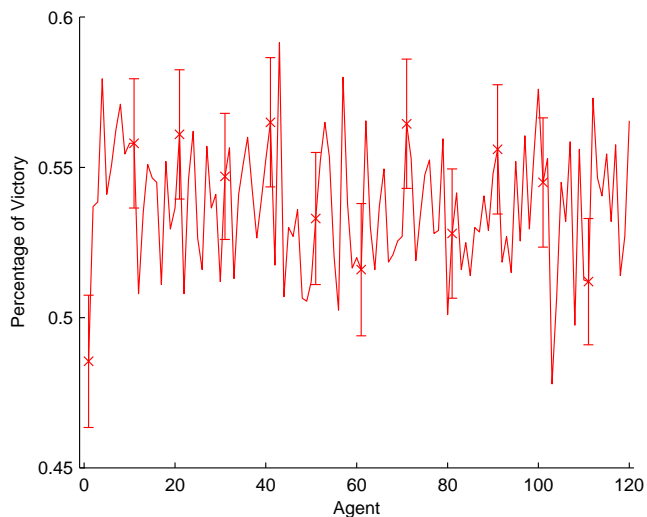


Figure 8: Learning graph in the not random order, as the algorithm tries to add each agent in the database.

lead to higher quality simulations, creating stronger players. It is notable that we could obtain a percentage of victory of around 59% against Fuego, in its default configurations for time limit and number of playouts per leaf.

5. DISCUSSION

In this paper we opened a new path for Computer Go: emergent behavior. In our approach, different agents play in the simulation phase of UCT Monte Carlo Go, which allows a greater diversity, increasing the quality of the simulations, and of the artificial player as a whole. It is possible to argue that other MCTS programs also have emergent behavior, as intelligent game play emerges from a playout strategy executed repetitively by a single agent. However, this work is the first to put Multi-Agent Systems and emergent behavior into perspective, showing new paths that can be explored to improve the current algorithms.

We could not achieve a significant percentage of victory against Fuego using the set of all possible 120 agents. However, we noticed that a selected set of agents could effectively improve the solution, and overcome Fuego. This inspired us to create a simple greedy learning algorithm, that tests if the presence of each agent contributes to improve the strength or not. With this algorithm, we could find a set of agents that won about 59% of the games. In the not random order, the first agents that the algorithm tried were already known to be good, and they were immediately selected. However, we had a set of 15 agents that we believed to be strong (when all of them were in the *agent database*, we obtained a percentage of victory of about 54%), and we were surprised when the learning algorithm reduced this set to only 5 agents. And also, the learning process increased the percentage of victory of our system in about 5%, compared to the solution that we could find manually. Therefore, it had a significant impact in our results.

However, even though we could significantly overcome Fuego with our agent set, it is still not so clear if the group performs better than the best agents, as the difference between them was small. As the number of possible combinations of agent sets is quite high, we believe there might be agent sets that perform even better, and might clearly overcome the best agents. Therefore, it is necessary to develop better algorithms for finding strong agent sets.

We believe that our approach is in a good direction to improve MCTS. However, even our straight $O(n)$ learning algorithm, executing in 104 cores, takes about 120 hours to finish. This happens because it is necessary to perform a great number of games in order to reach stable results, with low standard deviations. With the problems of sharing a cluster, like system maintenances, queues, machine reservation schedules, jobs being killed, etc, the whole execution took about one week and a half. Therefore, finding good agents is a difficult, computationally intensive problem.

Even though, we believe that much can still be discovered in that direction. An immediate future work is to regenerate the random list of agents, and run the learning process again. Would it select a similar set of agents? Could it discover an even stronger group? Another question that should be answered is the effect of adding not one agent to the database, but a set of agents. In other words, does each agent by itself contribute to the solution or is there improvement only when a specific set of agents are all together in the database? If so, how can that set be found? It is impos-

sible to test all combinations of agents. One idea could be to apply an algorithm like simulated annealing, and accept agents that decrease the solution, in order to escape local minima. In our experiments we could perceive that agents that perform bad individually are able to increase the quality of a certain set, so the effect of one agent might depend on the presence of other agents in the group. Unfortunately, it does not seem to be possible to apply learning algorithms like the evolutionary methods, due to the high cost of testing each solution.

Another possible future research path is to study how to apply Multi-Agent System paradigms in different ways. Our system employs a great number of agents during the simulations that are executed to evaluate the score of the leafs. It is possible to experiment with different applications of the paradigm. For example, what if different programs negotiate about a single move, like in [25]? How can we know which is the best movement among the ones suggested? In the case of Shogi the number of possible movements is more limited, and the convergence seems to be easier than in Go, allowing the application of simple majority voting algorithms. With the range of different possibilities allowed in a Go game, how can we solve the selection problem? Other possible direction is to try to apply Multi-Agent Systems in the tree search phase. Which algorithms could be applied? What benefits could we obtain? As can be seen, there is a great range of ideas and algorithms that can be inspired by this work.

6. CONCLUSION

In this paper we present a new paradigm to the state of the art of Go: Multi-Agent Monte Carlo Go. In our approach, different agents play in the simulation phase of UCT Monte Carlo Go, increasing the realism and the quality of the simulations by their emergent behavior. We could not achieve a significant result with all possible agents, but after selecting a good set of agents by a learning algorithm, we could significantly overcome the original system, Fuego. Therefore, we effectively increased the strength of UCT Monte Carlo Go. We present several discussions about our system, including directions for further improvement and points that should be better studied. We believe that our work presents a new paradigm for Monte Carlo Go, and it can be used as inspiration to a variety of different works.

We plan to further explore the research possibilities that were discussed in this paper. Therefore, our future work includes better exploring how to automatically learn good sets of agents. After running again our learning algorithm, but this time with different random agent lists, we plan to explore the effect of probabilistically accepting agents that decrease the solution, like simulated annealing algorithms. Other directions of research can be also explored, for example, discovering solutions of how to select the best move when different programs cooperatively play Go. We believe that much can still be researched, and Computer Go can be greatly improved by exploring Multi-Agent System techniques.

Acknowledgment

The authors would like to thank the Japanese Ministry of Education, Culture, Sports, Science and Technology for supporting this project.

7. REFERENCES

- [1] L. V. Allis. *Searching for Solutions in Games and Artificial Intelligence*. PhD thesis, University of Limburg, Maastricht, The Netherlands, 1994.
- [2] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47:235–256, May 2002.
- [3] D. B. Benson. Life in the game of go. *Inf. Sci.*, 10(2):17–29, 1976.
- [4] M. Boon. A pattern matcher for goliath. *Computer Go*, 13:13–23, 1990.
- [5] B. Bouzy. *Modelisation cognitive du joueur de Go*. PhD thesis, Université Paris, 1995. (in French).
- [6] B. Bouzy and T. Cazenave. Computer go: an ai oriented survey. *Artificial Intelligence*, 132:39–103, 2001.
- [7] B. Brugmann. Monte carlo go. Technical report, Physics Department, Syracuse University, 1993.
- [8] T. Cazenave. *Système d’Apprentissage par Auto-Observation. Application au Jeu de Go*. PhD thesis, Université Pierre et Marie Curie, 1996. (in French).
- [9] T. Cazenave and N. Jouandeau. A parallel monte-carlo tree search algorithm. In *CG ’08: Proceedings of the 6th international conference on Computers and Games*, pages 72–80, Berlin, Heidelberg, 2008. Springer-Verlag.
- [10] G. M.-B. Chaslot, M. H. Winands, and H. J. van den Herik. Parallel monte-carlo tree search. In *Proceedings of the 6th International Conference on Computer and Games*. Springer, 2008.
- [11] A. Colormi, M. Dorigo, and V. Maniezzo. Distributed optimization by ant colonies. In *European Conference on Artificial Life*, pages 134–142, 1991.
- [12] R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In H. J. van den Herik, P. Ciancarini, and H. H. L. M. Donkers, editors, *Computers and Games*, volume 4630 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 2006.
- [13] M. Enzenberger and M. Müller. Fuego - an open-source framework for board games and go engine based on monte-carlo tree search. Technical report, University of Alberta, Dept. of Computing Science, TR09-08, April 2009.
- [14] M. Enzenberger and M. Müller. A lock-free multithreaded monte-carlo tree search algorithm. In *Advances in Computer Games 12*, 2009.
- [15] K. J. Friedenbach, Jr. *Abstraction hierarchies: a model of perception and cognition in the game of go*. PhD thesis, University of California, Santa Cruz, 1980.
- [16] S. Gelly, J.-B. Hoock, A. Rimmel, O. Teytaud, and Y. Kalemkarian. On the Parallelization of Monte-Carlo planning. In *ICINCO*, Madeira Portugal, 2008.
- [17] S. Gelly, Y. Wang, R. Munos, and O. Teytaud. Modification of UCT with patterns in Monte-Carlo Go. Technical Report 6062, INRIA, France, November 2006.
- [18] H. Kato and I. Takeuchi. Parallel monte-carlo tree search with simulation servers. In *13th Game Programming Workshop (GPW-08)*, November 2008.
- [19] H. Kato and I. Takeuchi. Running ”zen” on computer clusters. In *14th Game Programming Workshop (GPW-09)*, November 2009. (in Japanese).
- [20] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948 vol.4, August 2002.
- [21] L. S. Marcolino and L. Chaimowicz. No robot left behind: Coordination to overcome local minima in swarm navigation. In *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, pages 1904–1909, 2008.
- [22] L. S. Marcolino and L. Chaimowicz. Traffic control for a swarm of robots: Avoiding group conflicts. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1949–1954, October 2009.
- [23] L. S. Marcolino and L. Chaimowicz. Traffic control for a swarm of robots: Avoiding target congestion. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1955–1961, October 2009.
- [24] M. Müller. *Computer Go as a Sum of Local Games: An Application of Combinatorial Game Theory*. PhD thesis, Zürich, 1995. (in French).
- [25] T. Obata, T. Sugiyama, K. Hoki, and T. Ito. Consultation algorithm in shogi: Can a set of players create a single strong player? In *14th Game Programming Workshop (GPW-09)*, November 2009. (in Japanese).
- [26] T. Oguri and Y. Kotani. Move decision method based on sds. In *14th Game Programming Workshop (GPW-09)*, November 2009. (in Japanese).
- [27] S. E. Page. *The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies*. Princeton University Press, January 2007.
- [28] C. W. Reynolds. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics (SIGGRAPH 87)*, pages 25–34. ACM Press, 1987.
- [29] Y. Soejima, A. Kishimoto, and O. Watanabe. Root parallelization of monte carlo tree search and its effectiveness in computer go. In *14th Game Programming Workshop (GPW-09)*, November 2009. (in Japanese).
- [30] T. Sugiyama, T. Obata, H. Saito, K. Hoki, and T. Ito. Consultation algorithm in brain game - a move decision based on the positional evaluation value of each player. In *14th Game Programming Workshop (GPW-09)*, November 2009. (in Japanese).
- [31] D. Wolfe. Go endgames are pspace-hard. *More Games of No Chance*, pages 125–136, 2002.
- [32] S.-J. Yen, C.-W. Chou, S.-C. Hsu, J.-C. Chen, and T.-N. Yang. Improvement of mcts in computer go. In *14th Game Programming Workshop (GPW-09)*, November 2009.

Towards a Unifying Characterization for Quantifying Weak Coupling in Dec-POMDPs

Stefan J. Witwicki and Edmund H. Durfee
Computer Science and Engineering
University of Michigan
Ann Arbor, MI 48109
{witwicki,durfee}@umich.edu

ABSTRACT

Researchers in the field of multiagent sequential decision making have commonly used the terms “weakly-coupled” and “loosely-coupled” to qualitatively classify problems involving agents whose interactions are limited, and to identify various structural restrictions that yield computational advantages to decomposing agents’ centralized planning and reasoning into largely-decentralized planning and reasoning. Together, these restrictions make up a heterogeneous collection of facets of “weakly-coupled” structure that are conceptually related, but whose purported computational benefits are hard to compare evenhandedly. The contribution of this paper is a unified characterization of weak coupling that brings together three complementary aspects of agent interaction structure. By considering these aspects in combination, we derive new bounds on the computational complexity of optimal Dec-POMDP planning, that together quantify the relative benefits of exploiting different forms of interaction structure. Further, we demonstrate how our characterizations can be used to explain why existing classes of decoupled solution algorithms perform well on some problems but poorly on others, as well as to predict the performance of a particular algorithm from identifiable problem attributes.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Theory, Performance

Keywords

Multiagent Planning, Coordination, Weak Coupling, Loose Coupling, Locality of Interaction, Policy Abstraction, Influence, Decentralized Markov Decision Processes, POMDPs

1. INTRODUCTION

The Decentralized Partially-Observable Markov Decision Process (Dec-POMDP) has emerged as a popular theoretical model for planning coordinated decisions for teams of agents

Cite as: Towards a Unifying Characterization for Quantifying Weak Coupling in Dec-POMDPs, S.J. Witwicki and E.H. Durfee, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 29-36.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

under uncertainty, but its well-established general NEXP hardness (reviewed by Goldman [6]) has raised concerns about its practical applicability beyond small toy problems. In response, researchers have defined a variety of subclasses amenable to efficient, scalable solution methods, but that impose various constraints on problem structure [2, 11, 15]. In this paper, we endeavor to illuminate significant aspects of Dec-POMDP interaction structure that make one problem easier than another, and to quantify the computational advantages of exploiting these aspects in concert.

Authors in the multiagent sequential decision making literature commonly use the terms “weakly-coupled” or “loosely-coupled” to classify problems involving agents whose interactions are limited (e.g., [8, 12, 13, 22]). Intuitively, weakly-coupled interaction structure engenders conditional independencies among individual agents’ decisions that allow for efficient decomposition of joint planning and reasoning. However, different authors’ uses of these terms refer to slightly different structural conditions, and often frame the consequences of the structure in slightly different algorithmic contexts. Given the heterogeneity of structural conditions for weak coupling, and the diverse contexts of published results, it is difficult to ascertain the computational advantages of the various structures in relation to one another.

Here, we generalize and synthesize several elements of problem structure into a more unified characterization of weak coupling. In particular, we highlight three complementary aspects of weakly-coupled problem structure: *agent scope size*, *state factor scope domain size*, and *degree of influence*. Not only does our characterization highlight useful relationships between these three aspects, but it also concretely quantifies the relative computational benefits of exploiting each. By considering these three aspects in concert, we derive new bounds on the worst-case complexity of optimal Dec-POMDP planning (the context of which we describe in Section 2). After presenting our characterization and theoretical results in Section 3, we illustrate the usefulness of this contribution in Section 4 by demonstrating that our theory helps (1) to better explain trends observed in past work, and (2) to predict the performance of solution algorithms based on the degree to which test problems are weakly coupled. As a case study, we illustrate how our theoretical results can be used in conjunction with empirical analysis to extrapolate the relative performance of a particular algorithm, Optimal Influence-space Search [22], on 4-agent problems. In Section 5, we relate our characterization to foundational and alternative analyses from past work, and conclude with a discussion of our results and future work in Section 6.

2. CONTEXT

We are considering the problem of planning optimal policies for a group of agents whose behavior is modeled as a Dec-POMDP. To illustrate the interaction structure that we are characterizing, we refer to several examples expressed using a particular Dec-POMDP model called a Transition-Decoupled POMDP (TD-POMDP) [22], whose specification emphasizes conditional independencies among agents.

Figure 1a shows a simple problem involving a three-agent team whose objective is to plan the coordinated executions of *tasks*. Here, task interdependencies called *enablements* each represent the constraint that one task must have completed with positive outcome quality before another task can begin. The execution of each task, by the agent that owns that task, is also constrained by a *window* that expresses the task’s earliest start time and latest finish time (after which the task will “fail” by achieving zero outcome quality). The uncertainty of each task’s execution is conveyed by its *outcome distribution*, which assigns a probability (P) to each possible duration (D) and quality (Q). In the past, Dec-POMDP researchers have studied examples of this flavor that were geared towards application domains such as Mars rover control [13, 22] and disaster response [11].

2.1 Decentralized POMDP

A (finite-horizon) Dec-POMDP [1, 6, 16, 22] is specified with a tuple $\mathcal{M} = \langle \mathcal{N}, S, A, \Omega, O, R, P, T \rangle$, where \mathcal{N} is a set of n agents, S is a finite set of world states, with a distinguished initial state, and $A = \times_{i \in \mathcal{N}} A_i$ is the joint action space, each component of which refers to the set of actions of an agent $i \in \mathcal{N}$. The transition function $P(s'|s, a)$ specifies the probability distribution over next states given that joint action $a = \langle a_1, a_2, \dots, a_n \rangle \in A$ is taken in state $s \in S$. The reward function $R(s, a, s')$ specifies the immediate value of taking joint action a in state s and arriving in state s' . Observation function $O(o'|a, s')$ specifies the probability of the joint observation $o \in (\Omega = \times_{i \in \mathcal{N}} \Omega_i)$, observed upon taking a and transitioning into s' .

The team’s behavior is specified with a *joint policy* $\pi = \langle \pi_1, \dots, \pi_n \rangle$, where each component π_i (agent i ’s *local policy*) maps agent i ’s observation history δ_i^t to an action a_i , thereby encoding a deterministic decision rule for any sequence of observations that each agent might encounter. The set of possible joint policies Π denotes the *policy space*, and Π_i agent i ’s local policy space. The *value* $V(\pi)$ of a joint policy $\pi \in \Pi$ is the expected cumulative reward received by the team (from times 1 to T) when executing that policy from the initial state. Finally, Dec-POMDP planning is the problem of computing the *optimal joint policy* π^* , which can be expressed as a maximization: $\pi^* = \arg \max_{\pi \in \Pi} V(\pi)$.

2.2 Transition-Decoupled POMDP

The Dec-POMDP is an extremely general representation of joint behavior, allowing for arbitrary dependencies among agents’ observations and action consequences. As such, the naïve Dec-POMDP specification is oblivious to any interaction structure that may exist among agents. To express some of the structure (which we find useful in Section 3) that is present in problems like the one depicted in Figure 1a, we turn to the TD-POMDP model [22].

The TD-POMDP assumes an inherent decomposition of the joint model into transition-dependent local agent models. The world state s is *factored* into individual state features

that are distributed among agents’ *local states* $\langle s_1, \dots, s_n \rangle \in S = \times_{i \in \mathcal{N}} S_i$; however, each feature does not necessarily reside exclusively in a single agent’s local state s_i . Figure 1b depicts the decomposition of world state for the example problem as a two-stage dynamic Bayesian network, wherein feature “Den” (encoding whether or not Task D is enabled) is shared among agent 1’s and agent 2’s local states, and *time* is shared among all agents’ local states. These features are referred to as *mutually-modeled features*, comprising set \bar{m} .

Similarly, the TD-POMDP specifies decomposable observation functions and decomposable local rewards [22]. In this paper, we consider TD-POMDP problems for which the joint value function decomposes into local value functions $V(\pi) = \sum_{i \in \mathcal{N}} V_i(\pi)$, each of which, $V_i(\pi)$, is equal to one agent i ’s expected cumulative *local* reward. In the example problem, local rewards account for the quality accrued whenever an agent finishes one of its tasks. The TD-POMDP explicitly distinguishes features in an agent j ’s local state that are controlled by agent i as a *nonlocal features* \bar{n}_j , wherein each serves as an attribute through which i can affect j ’s local state, observations, and subsequent local transitions [22]. For instance, in Figure 1b, when agent 1 completes task C, nonlocal feature Fen (F enabled) changes from *false* to *true*, thereby altering how agent 3’s actions can affect feature F .

A TD-POMDP’s transition-dependent interactions may be illustrated graphically using an *agent interaction digraph* [22], examples of which are shown in Figure 2. The interaction digraph contains a vertex for each agent, and an edge for each nonlocal feature that connects the controlling agent with the affected agent. For any two agents i and j , there may be more than one edge leading from i to j , one for each nonlocal feature controlled by i and affecting j . For the purpose of our analysis we shall denote the *digraph ancestors* of an agent j as $\Lambda_j = \{i \neq j \mid \text{there is a directed path from } i \text{ to } j\}$, and the set of *digraph descendants* of agent i as $\Psi_i = \{j \neq i \mid \text{there is a directed path from } i \text{ to } j\}$. In contrast, we shall use the word *peer* to refer to “some other agent” in \mathcal{N} without the implication of any particular graphical relationship.

2.3 Relationship Between Dec-POMDP Planning and Constraint Optimization

Our analysis in Section 3 makes use of a reformulation of the Dec-POMDP planning problem into a *constraint optimization problem* (COP). The reformulation is a slight generalization of that explored in past work [2, 15]. In review, a classical COP [4] is specified as a tuple $\mathcal{C} = \langle X, D, C \rangle$, where $X = \{x_1, \dots, x_n\}$ is a set of n *variables* with possible assignments $\bar{a} = \langle a_1, \dots, a_n \rangle \in D = \{D_1, \dots, D_n\}$, and C is a set of *constraints*. Each constraint represents a cost function C_k with a restricted (variable) scope $Q_k \subseteq \{1, \dots, n\}$, such that $C_k : [\times_{i \in Q_k} D_i] \mapsto \{\mathbb{R}, \infty\}$. The restricted scopes of COP constraints constitute graphical structure that is naturally expressed using a *constraint graph* \mathcal{G} . Illustrated in Figure 2, there is a hyperedge for each constraint C_k that connects those vertices (which we refer to as *neighbors*) corresponding to the variables indexed by Q_k .

In solving a COP, and obtaining solution \bar{a}^* , the objective is to minimize the summation of cost values of the variable assignments: $\bar{a}^* = \arg \min_{\bar{a}} \sum_{k=1}^{|C|} C_k(\bar{a})$. Analogously, the objective of Dec-POMDP planning is to maximize the expected utility of the joint policy, which can often be decomposed into component value functions [11, 15, 16, 22], one for each agent in the case of the TD-POMDP.

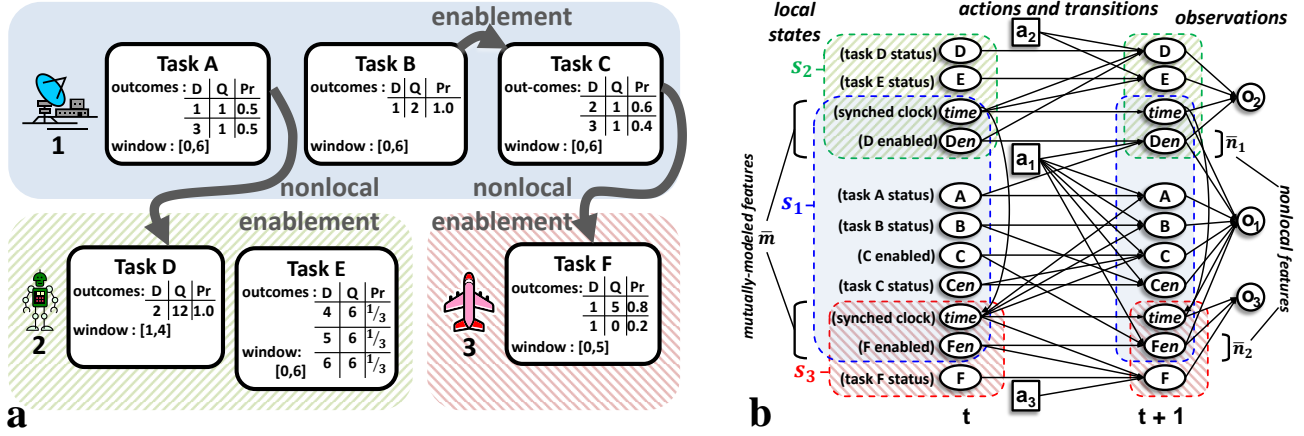


Figure 1: An example problem (a) and corresponding TD-POMDP specification (b).

OBSERVATION 1. A Dec-POMDP \mathcal{M} , whose value function $V(\pi = \langle \pi_1, \dots, \pi_n \rangle)$ decomposes into a summation of component value functions $V(\pi) = \sum_k V_k(\pi)$, reduces to a COP $\mathcal{C}_{\mathcal{M}} = \langle X, D, C \rangle$ structured as follows:

- X contains exactly one variable x_i for each agent i ,
- the domain of x_i is agent i 's local policy space: $D_i \equiv \Pi_i$,
- an assignment $\bar{a} = \langle \pi_1, \dots, \pi_n \rangle$ is a joint policy,
- C consists of a single constraint $C_k(\bar{a} \equiv \pi) = -V_k(\pi)$ for each component V_k of \mathcal{M} 's value function,
- and the solution \bar{a}^* is an optimal joint policy π^* .

This reformulation gives rise to a decoupled joint policy search methodology (such as that employed in past work [2, 14, 22]). In contrast to centralized methods that optimize all components of the optimal joint policy at once, a decoupled joint policy search is an iterative process where, at each step, an each agent i computes one possible local policy, $\pi_i^*(\bar{\pi}_{\neq i}) = \arg \max_{\pi_i} V(\pi_i, \bar{\pi}_{\neq i})$, referred to as a **best response** to proposed local policies $\bar{\pi}_{\neq i}$ of i 's peers. Equation 1 below reduces the computation of an optimal joint policy for a three agent Dec-POMDP problem (such as in Figure 1) to a series of best response calculations.

$$\begin{aligned}
 \pi^* &= \arg \max_{\langle \pi_1, \pi_2, \pi_3 \rangle} V(\pi_1, \pi_2, \pi_3) \\
 &= \arg \max_{\langle \pi_1, \pi_2 \rangle} V(\pi_1, \pi_2, \pi_3^*(\pi_1, \pi_2)) \\
 &= \arg \max_{\pi_1} V(\pi_1, \pi_2^*(\pi_1), \pi_3^*(\pi_1, \pi_2^*(\pi_1))) \\
 &\quad (\text{where } \pi_2^*(\pi_1) = \arg \max_{\pi_2} V(\pi_1, \pi_2, \pi_3^*(\pi_1, \pi_2)))
 \end{aligned} \tag{1}$$

The search implied by the $\arg \max$ invocations in Eq. 1, which enumerates all combinations of local policies, serves as the basis for more advanced decoupled solution methods cited in the next section.

3. DIMENSIONS OF WEAKLY COUPLING

Intuitively, the computational benefit of solving a problem with a decoupled solution method instead of a centralized method depends upon the presence of problem structure that renders agents more or less independent of each other. Here, we generalize and formally characterize three different previously-studied aspects of weakly-coupled problem structure whose exploitation has been shown to be beneficial in

past work (which we review in Section 5). Our characterization takes the form of a three-dimensional landscape that can be used to quantify the advantage gained through exploiting a problem's interaction structure and, ultimately, to predict the amount of computation needed to solve the problem. We describe each dimension in Sections 3.1–3.3, over the course of which we gradually refine a bound on the computational complexity of Dec-POMDP planning, and then in Section 3.4 we bring these terms together into a unified characterization.

3.1 Agent Scope Size

The first aspect that we examine lies in the graphical structure present in a Dec-POMDP \mathcal{M} 's equivalent COP constraint graph $\mathcal{G}_{\mathcal{M}}$. The connectivity of each hyperedge is dictated by the scope Q_k of a constraint C_k , which is equal to the *agent scope* [7, 16] of component value function V_k .

Definition 1. The **agent scope**, denoted Q_k , of a (component) value function $V_k()$ is the subset of agents on whose policies its value depends, such that $V_k : [\times_{i \in Q_k} \Pi_i] \mapsto \mathbb{R}$.

In a TD-POMDP agent, for instance, the scope Q_i of an agent i 's local value function contains all agents that can affect i 's rewards through their actions, which are i and its interaction digraph ancestors: $Q_i = \{i\} \cup \Lambda_i$ [21]. In a Network-Distributed POMDP (ND-POMDP), the size of the agent scope corresponds to *local neighborhood size* [15].

At one extreme of the weak coupling spectrum, agents are *uncoupled*: they do not interact, so the constraint graph consists of n unconnected vertices. In this case, the optimal joint policy is simply the combination of independently-computed optimal local policies: $\pi^* = \langle \arg \max_{\pi_i} V_i(\pi_i), \forall i \in \mathcal{N} \rangle$, and agent scope $Q_i = \{i\}, \forall i$. At the opposite extreme, all agents' decisions are affected by all other agents, and hence no agent can optimize its local policy without considering the potential policies of all other agents. In between these two extremes, there exist conditional independencies that allow agents to plan independently of some peers but not others.

Definition 2. An agent i is **conditionally decision-independent** of agent j conditioned on peer agents $K \subseteq (\mathcal{N} - \{i, j\})$ if: $\forall \{\pi_j^x, \pi_j^y\} \subseteq \Pi_j, \forall \bar{\pi}_K \in (\times_{k \in K} \Pi_k)$, $\arg \max_{\pi_i \in \Pi_i} V(\pi_i, \pi_j^x, \bar{\pi}_K) = \arg \max_{\pi_i \in \Pi_i} V(\pi_i, \pi_j^y, \bar{\pi}_K)$.

In the example from Figure 1, agents 2 and 3 are *conditionally decision-independent* of each other conditioned on

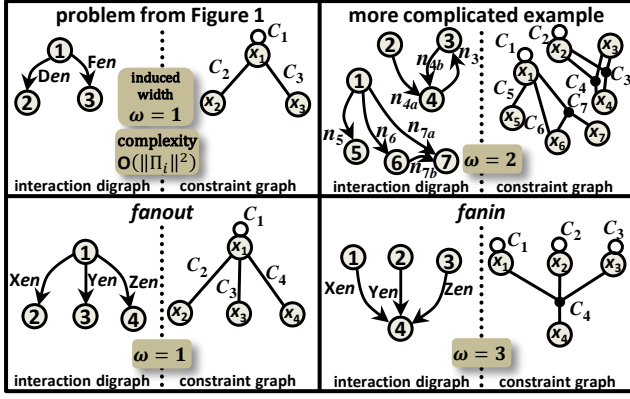


Figure 2: Examples of constraint graphs (right) derived from TD-POMDP interaction digraphs (left).

agent 1. Intuitively, this is because agent 2 cannot affect the values of agent 3’s local state features nor *vice versa*, no matter what actions they decide to take. Mathematically, it is because, from the constraint graph in Figure 2 (upper-left), agent 3’s best response $\pi_3^*(\pi_1, \pi_2)$ is independent of π_2 :

$$\begin{aligned} \pi_3^*(\pi_1, \pi_2) &= \arg \max_{\pi_3} V(\pi_1, \pi_2, \pi_3) \\ &= \arg \max_{\pi_3} [V_1(\pi_1) + V_2(\pi_1, \pi_2) + V_3(\pi_1, \pi_3)] \\ &= \arg \max_{\pi_3} V_3(\pi_1, \pi_3) \equiv \pi_3^*(\pi_1) \end{aligned}$$

and *vice versa*. The practical benefit of this *conditional decision independence* relationship is a simpler optimal solution computation. Substituting $\pi_3^*(\pi_1)$ (as well as the equivalently-reduced $\pi_2^*(\pi_1)$) into Equation 1 significantly reduces the combinations of local policies that need be considered:

$$\pi^* = \arg \max_{\pi_1} V(\pi_1, \pi_2^*(\pi_1), \pi_3^*(\pi_1)) \quad (2)$$

Whereas Equation 1 required $\|\Pi_1\| \|\Pi_2\| \|\Pi_3\|$ evaluations of $V()$, Equation 2 requires only $(\|\Pi_1\| \|\Pi_2\| + \|\Pi_1\| \|\Pi_3\| + \|\Pi_1\|)$ evaluations due to the reduction in the number of best responses, given the restricted *agent scopes* of $V_2()$ and $V_3()$.

Using COP theory, we can generalize the computational reduction as well as the methodology of exploiting graphical structure beyond our simple example problem. The computation performed in Equation 2 is an instance of *bucket elimination* (essentially, nonserial dynamic programming) [4], a general solution methodology that performs well on sparsely-connected constraint graphs. Dechter has proven that the worst-case time and space complexity of bucket elimination is $O(\|C\| \|D_i^{max}\|^{\omega+1})$, where $\|C\|$ is the number of cost functions, D_i^{max} is the largest domain size of any COP variable and ω is the *induced width* of the constraint graph [4].

OBSERVATION 2. *The worst-case time and space complexity of optimal planning for Dec-POMDP \mathcal{M} is bounded by $O(n \cdot \|\Pi_i^{max}\|^{\omega+1})$, where n is the number of component value functions, Π_i^{max} is the largest local policy space, and ω is the induced width of the equivalent constraint graph $\mathcal{G}_{\mathcal{M}}$.*

By Observation 2, a lower induced width implies an exponential reduction in worst-case computation time. However, note that the local policy space size $\|\Pi_i^{max}\|$, at the base of the exponent, is itself exponentially dependent on the number of local observations histories: $\|\Pi_i^{max}\| = O(\|A_i\|^{\|O_i\|^T})$ [14].

By Dechter’s definition [4], the induced width ω may be calculated by taking the minimum, over all possible orderings of vertices in $\mathcal{G}_{\mathcal{M}}$, of the following measure: process vertices in order from last to first, for each vertex connecting its earlier-ordered neighbors, then return the largest number of earlier-ordered neighbors of any vertex. Alternatively, we can estimate ω using *agent scope size*. While in general, $\omega \geq (\max_k \|Q_k\| - 1)$ (which follows from the definitions of ω and Q_k), for a wide variety of TD-POMDP interaction digraph topologies (some of which are shown in Figure 2), $\omega = (\max_k \|Q_k\| - 1)$.

3.2 State Factor Scope Domain Size

The theoretical results presented thus far assume a naïve algorithm for performing best response calculations: enumeration of all local policies $\pi_i \in \Pi_i$ and explicit evaluation of $V(\pi_i, \bar{\pi}_{\neq i})$ for each. In a classical COP, enumeration of variable domains would be the only way to compute a best response. However, the COP that we are solving involves policy variables with structured domains. To exploit this structure, several algorithms have been developed for computing a best response by solving a single-agent POMDP model seeded with peers’ policy information [14, 15, 22]. Aside from harnessing the efficiencies of state-of-the-art POMDP solvers, a best-response model can also exploit weakly-coupled problem structure. In particular, a best-response model does not necessarily need to represent all world state features [15, 22].

Intuitively, there may be features that have no bearing on the value ascribed to the agent’s own behavior. For instance, in Figure 1b, the enabling of Task F (encoded by feature *Fen*, appearing in agent 1’s local state, but unobservable to agent 2) is inconsequential to agent 2 as it plans its best response policy $\pi_2^*(\pi_1)$. Using Definition 3, which we have adapted from previous work [7, 16] to fit this context, feature *Fen* is not in agent 2’s state factor scope.

Definition 3. An agent i ’s **state factor scope** \mathcal{X}_i is the minimal¹ set of features sufficient for modeling the (belief) state used to compute i ’s optimal best response.

Becker *et al.* have derived that a Transition-Independent Dec-MDP (TI-Dec-MDP) agent i ’s best response may be calculated with an *augmented MDP* whose state space includes *only* i ’s local state features (but whose rewards are modified to account for peer agent j ’s proposed policy) [2]. In this case, even though the joint utility is dependent upon features from both agents’ local state representations, it suffices for agent i to reason over a greatly-reduced space of features when computing a best response. In earlier work [22], we have developed a POMDP for computing the best response for a TD-POMDP agent i that includes (at most) the features from i ’s local state s_i and the histories of mutually-modeled features \bar{m}_i . A TD-POMDP agent i ’s state factor scope is thus $\mathcal{X}_i \subseteq \{s_i, \bar{m}_i\}$.

Intuitively, the smaller the portion of the world state that an agent observes and interacts with, the smaller its state factor scope, and the easier its local planning and reasoning becomes. Accounting for the sizes of the domains of features in the state factor scope, denoted $Dom(\mathcal{X}_i)$, we can refine our bound on computational complexity as follows.

¹Given that multiple flavors of best response model may be applicable [14, 22], we are most interested in those that exploit weakly-coupled problem structure by reducing their modeled set of features as much as possible.

THEOREM 1. *The worst-case complexity of Dec-POMDP planning is $O(n \cdot \text{EXP}(\| \text{Dom}(\mathcal{X}_i^{\text{max}}) \|) \cdot \|\Pi_i^{\text{max}}\|^\omega)$, where $\| \text{Dom}(\mathcal{X}_i^{\text{max}}) \| = \max_{i \in \mathcal{N}} \| \text{Dom}(\mathcal{X}_i) \|$.*

PROOF SKETCH. The derivation of the complexity result from Observation 2 entails every best response computation requiring an $\arg \max_{\pi_i}$ to be taken, enumerating the local policy space bounded by $\|\Pi_i^{\text{max}}\|$, for all combinations of policies of ω peers, yielding complexity $\|\Pi_i^{\text{max}}\| \|\Pi_i^{\text{max}}\|^\omega$ for each of the n agents. By replacing each best response calculation with one POMDP solution, we can substitute the first term $\|\Pi_i^{\text{max}}\|$ in our complexity computation with the complexity of solving a finite-horizon POMDP, which is $O(\text{EXP}(\|S\|) = \text{EXP}(\| \text{Dom}(\mathcal{X}_i^{\text{max}}) \|))$ given that the state space is bounded by $\| \text{Dom}(\mathcal{X}_i^{\text{max}}) \|$ [21]. \square

3.3 Degree Of Influence

Making use of Definition 2, for any two agents i and j , i is either decision-dependent on j or decision-independent of j (possibly conditioned on some other agents). Considering the rich space of dependencies that may exist between the two agents, a binary relation such as *decision-independent* lacks the precision to characterize weakly-coupled problems satisfactorily. For instance, in the example problem from Figure 1a, agent 2 is decision-dependent on agent 1, but only dependent on those decisions relating to the execution of Task A. Whether agent 1 executes Task B after completing Task A, or simply idles, cannot impact agent 2's decisions in any way. Moreover, any two of agent 1's possible policies, π_1^x and π_1^y that differ only in the decisions made after completing Task A induce the same best response from agent 2.

Definition 4. Two policies, π_i^x and π_i^y , of agent i are **impact-equivalent**, denoted $\pi_i^x \stackrel{I}{\equiv} \pi_i^y$, if π_i^x and π_i^y result in the same peer best responses: $\forall j \neq i, \forall \bar{\pi}_{(K=\mathcal{N}-\{i,j\})}$, $\pi_i^x \stackrel{I}{\equiv} \pi_i^y \Leftrightarrow \left[\arg \max_{\pi_j} V(\pi_i^x, \pi_j, \bar{\pi}_K) = \arg \max_{\pi_j} V(\pi_i^y, \pi_j, \bar{\pi}_K) \right]$.

Definition 5. An **impact equivalence class** $E_{i,x}$ is a set of impact-equivalent policies: $\forall \{\pi_i^x, \pi_i^y\} \in E_{i,x}, \pi_i^x \stackrel{I}{\equiv} \pi_i^y$.

In essence, an agent i 's local policy space can be partitioned into disjoint equivalence classes, each of which may impact other agents in the system in a different way, thereby (potentially) inducing a different combination of best responses from i 's peers. Definitions 4–5 elicit a spectrum of varying degrees of agent dependence. At one end of the spectrum, all of agent i 's policies are grouped into a single impact equivalence class, indicating that any given peer j 's behavior is unaffected by i 's decisions. At the opposite end of the spectrum, agent j 's best response is highly sensitive to the policy that i adopts, such that no two policies of agent i are impact-equivalent, and the number of i 's impact equivalence classes is equal to the size of its policy space $\|\Pi_i\|$.

There are several Dec-POMDP planning algorithms that take advantage of this kind of weak agent coupling [2, 22]. Each algorithm employs what we shall call a *partitioning scheme* that implicitly partitions each agent i 's local policy space into a set of impact equivalence classes $\mathbb{P}_i = \{E_{i,x}\}$, parameterizing each class with information representative of the policies from that class. The key is that to compute the optimal joint policy, it suffices for each agent j to compute a best response to just one of the local policies from each of

agent i 's $\|\mathbb{P}_i\|$ impact equivalence classes. This set of best responses is referred to as agent j 's *optimal coverage set* [2] because it sufficiently covers every possible policy of agent i .

Definition 6. For a given problem, the **degree of influence** $d_{\mathbb{P}}$, afforded by a partitioning \mathbb{P} , is the maximum ratio of impact equivalence classes to local policies:

$$d_{\mathbb{P}} = \max_{i \in \mathcal{N}} \frac{\|\mathbb{P}_i\|}{\|\Pi_i\|}. \quad (3)$$

By Definition 6, if a very coarse partitioning is found for a particular problem, wherein partitions contain large numbers of local policies, a low *degree of influence* $d_{\mathbb{P}}$ has been achieved. All else being equal, problems with a low degree of influence should be easier to solve than problems with a high degree of influence because of the reduction, from $\|\Pi_i\|$ to $\|\mathbb{P}_i\|$, in the number of necessary best responses per step of a distributed joint policy search. However, the computational benefit of a coarse partitioning could be offset by the computational overhead required to partition each agent's local policy space, whose worst case we denote $C_{\mathbb{P}}$.

3.4 Unified Characterization

In the past three subsections, we have quantified three problem characteristics associated with the degree of coupling. Conceptually, *agent scope size* refers to the number of agents in the system that are affecting each others' decisions, *state factor scope domain size* refers to the portion of state feature values that must be considered by each individual agent when coordinating its decisions, and *degree of influence* refers to the proportion of unique ways that agents can impact each others' decisions (subject to a given partitioning scheme). Each aspect manifests itself in a different set of problem attributes, and each affects the overall character in a different manner. However, we can formally characterize the combination of their effects as follows:

THEOREM 2. *The worst-case time and space complexity of Dec-POMDP planning, using a decoupled solution method that partitions agents' policy spaces, is bounded by:*

$$O(n \cdot \text{EXP}(\| \text{Dom}(\mathcal{X}_i^{\text{max}}) \|) \cdot (d_{\mathbb{P}} \|\Pi_i^{\text{max}}\|)^\omega + n \cdot C_{\mathbb{P}}) \quad (4)$$

where n denotes the number of component value functions, $\text{Dom}(\mathcal{X}_i^{\text{max}})$ denotes the largest domain of any agent's state factor scope (Def. 3), $d_{\mathbb{P}}$ denotes the degree of influence (Def. 6) given partitioning \mathbb{P} , $C_{\mathbb{P}}$ is the worst-case complexity of computing \mathbb{P} , Π_i^{max} denotes the largest local policy space, and ω is the induced width.

PROOF SKETCH. Equation 4 is straightforwardly derived by manipulating the bound from Theorem 1. The base of the exponent is replaced with $d_{\mathbb{P}}$ due to the worst-case reduction in the number of best response computations afforded by \mathbb{P} . Next, a second term is added accounting for the accumulation of computation required by the partitioning process. \square

Parameters $\{\omega, \mathcal{X}_i^{\text{max}}, d_{\mathbb{P}}\}$ can be thought of as separate dimensions whose combination provides a concrete measure of the weakness (or strength) of coupling of a problem. A problem's worst-case complexity depends on where it lies along the spectra of *agent scope size*, *state factor scope domain size*, and *degree of influence*. For any two problems, we can now compare their worst case complexities by estimating the values of the three parameters and positioning each in the

3-dimensional space. Given a factored representation of Dec-POMDP problem structure (such as a TD-POMDP [22], ND-POMDP [15], or a more general factored Dec-POMDP [16]), the first two parameters, ω and \mathcal{X}_i^{\max} , can be evaluated directly from the problem specification. The third dimension, $d_{\mathbb{P}}$, is not readily assessable from any Dec-POMDP problem specification that we are aware of. Moreover, $d_{\mathbb{P}}$ is inherently tied to the partitioning scheme used by the solution algorithm. Thus, for any given algorithm, we propose to estimate $d_{\mathbb{P}}$ through empirical profiling of the partitioning scheme (as we demonstrate in Section 4.2).

4. EVALUATION

A significant contribution of the theory we presented in Section 3 lies in its explanatory and predictive power, a claim which we now defend (anecdotally in Section 4.1, and empirically in Section 4.2).

4.1 Explaining Trends

Researchers have developed a number of different algorithms for exploiting the kinds weakly-coupled problem structured formalized above [2, 9, 10, 13, 14, 15, 16, 22]. Our unified characterization can explain some of the trends observed in the performance of these algorithms that are not easily explained without considering combinations of dimensions of agent coupling.

For instance, the successes of a family of ND-POMDP algorithms [9, 15] in scaling to many agents has been attributed to the reduced agent scope associated with ND-POMDP agents' *local neighborhoods* [9, 10, 15]. That is, as long as the agent scope remains small, these algorithms are expected to be practical. However, a generalized version of one of these algorithms (JESP [14]) has recently been reported as intractable for a test set of Distributed POMDPs with Coordination Locales (DPCLs) containing *just two agents*, even when generating an approximate solution [20]. A likely explanation for this phenomenon is contained within Equation 4, which suggests that it was not the agent scope of the problems that foiled JESP but instead the cost of JESP's best response calculation. Whereas ND-POMDP problems have an inherently restricted state factor scope due to the strict separation of transition-independent agents' local states, DPCL problems involve transition-dependent agents that need to reason about each others' state variables in order to compute optimal best responses (which JESP employs in computing approximate solutions), making the DPCL more strongly coupled even in its two-agent incarnation.

Transition dependence alone does not make a problem strongly coupled, however. In earlier work, we demonstrated the capability of an algorithm inspired by JESP, Optimal Influence-Space search (OIS), to scale optimally to sets of TD-POMDP problems with four transition-dependent agents [22]. At the time, little was understood about the structure of these problems that OIS was exploiting, especially given the wide range of OIS runtimes reported for a single set of problems. The theory developed in Section 3 leads us to attribute the successes of OIS to a low degree of influence in the test problems, a claim that is supported by empirical evidence presented in Section 4.2.1.

4.2 Predicting Performance of OIS

Our theory can also be used to make detailed predictions about the computational overhead of algorithms such

as OIS that exploit weakly-coupled problem structure. Instead of presenting a comprehensive empirical analysis of all the dimensions of weak coupling, we use the limited space here to illustrate how to use Equation 4 to predict the relative computation time taken by OIS to solve variations of two example problems, named *fanout* and *fanin*, whose interaction digraphs are shown in Figure 2 (bottom). The two examples differ in their topology, but both include four agents connected by three *enablement* features, each linking randomly-selected tasks from the task sets of the corresponding agents. Each agent's task set contains three tasks, each with three randomly-selected outcomes whose *qualities* are random integers $\in (1, 10)$ and whose *durations* are random integers $\in (1, 5)$ selected without replacement.

Aside from demonstrating how to make empirically-guided theory-driven predictions, this experiment serves to elucidate the relationship between the *window* sizes of agents' tasks (examples of which appear in Figure 1a) and the computation time of OIS that we observed in an earlier analysis [22]. As such, we have generated sets of problems (of both the *fanin* and *fanout* flavors) whose task window sizes were fixed at $\{1, 2, 3, 4, \text{ and } 5\}$ and whose earliest start time and latest end times were selected so as to position the task's fixed-size window uniformly randomly in the interval $(0, 5)$.

4.2.1 Profiling Partitioner and Best-Response Solver

Whereas Equation 4 conveys a general bound, OIS employs a specific form of impact equivalence partitioning. In order to estimate the degree of influence $d_{\mathbb{P}}$ and partitioning complexity $C_{\mathbb{P}}$ that OIS will achieve on *fanin* and *fanout*, we profile OIS's partitioner and best response solver on two sets of smaller 2-agent enabler-ablee problems, one in which the enabler controls a single enablement (as do the enablers in *fanin*), and one in which the enabler controls three enablements (as does the enabler in *fanout*).

Figure 3 (top-left) shows $d_{\mathbb{P}} \|\Pi_i^{\max}\|$ plotted as a function of *window size*. Each point represents the mean value of 20 randomly-generated profiling problems. Both flavors of problems exhibit an exponential trend in the number of equivalence classes, which explains why, in previously-reported results [22], OIS appeared to compute optimal solutions in exponentially less time as the window size was decreased. However, a striking feature of these plots is the high variance² (indicated by the error bars). This suggests that, in addition to window size, there are other factors at play that have a significant effect on the degree of influence. We ran an additional experiment in which we held the window size constant at 3 and varied the *earliest start time* of one enabling task, the results of which are shown in Figure 3 (top-right). From this plot, it appears that the temporal placement of agents' interactions is also a good predictor of the degree of influence, though analysis beyond the scope of this paper is required to verify this supposition.

Upon measuring the computation time taken by the enabler to perform its impact equivalence partitioning (represented as $C_{\mathbb{P}}$ in Eq. 4), we observed that it did not consume more than 70 percent of the total solve time on any given problem. We thereby deduced that for this particular suite of problems, the first term of Equation 4, $n \cdot \text{EXP}(\|\text{Dom}(\mathcal{X}_i^{\max})\|) \cdot (d_{\mathbb{P}} \|\Pi_i^{\max}\|)^{\omega}$, would be just as strong a predictor of OIS's

²To verify that the means were not simply driven up by outliers, we also examined the medians (not shown here), and observed the same exponential trend.

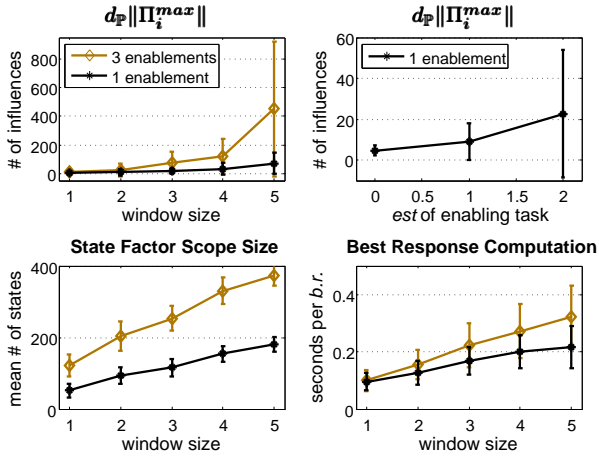


Figure 3: Profiling: $d_{\mathcal{P}}\|\Pi_i^{max}\|$ (top-left), $d_{\mathcal{P}}\|\Pi_i\|$ vs. earliest start time (top-right), $\|Dom(\mathcal{X}_i^{max})\|$ (bottom-left), and best-response comp. time (bottom-right).

relative computation time as the sum of both terms.

Whereas $\text{EXP}(\|Dom(\mathcal{X}_i^{max})\|)$ in Equation 4 is a worst-case bound on the complexity that makes no assumptions about the POMDP solver, OIS uses a particular solver whose computation time we measured on the problems in our profiling set. Figure 3 (bottom-left) plots the state factor scope domain size ($\|Dom(\mathcal{X}_i)\|$), measured as the actual size of the state space of the best-response model, as well as the mean computation time per best response (bottom-right). Note that, due to their topologies, the “3 enablement” problem profiles the best response computation of *fanin*, whereas the “1 enablement” problem profiles that of *fanout*. For both flavors, we observe a slight increase in both state factor scope domain size and best response computation time (which for these problems appear to be linearly, not exponentially, related in practice).

4.2.2 Predicting Relative Computation Time

Next, we evaluated $n \cdot \text{EXP}(\|Dom(\mathcal{X}_i^{max})\|) \cdot (d_{\mathcal{P}}\|\Pi_i^{max}\|)^\omega$ for both *fanin* and *fanout* across all window sizes, replacing $\text{EXP}(\|Dom(\mathcal{X}_i^{max})\|)$ with the profiled best response computation time shown in Figure 3 (bottom-right) and $d_{\mathcal{P}}\|\Pi_i^{max}\|$ with its profiled value (top-left). For *fanin* problems, ω was set to 3, and for *fanout*, 1; for all problems, $n = 4$. Figure 4 shows the computation time predicted by our theory (left), and the mean values of actual runtime of OIS (right) on 20 random *fanin* and *fanout* problems per window size.

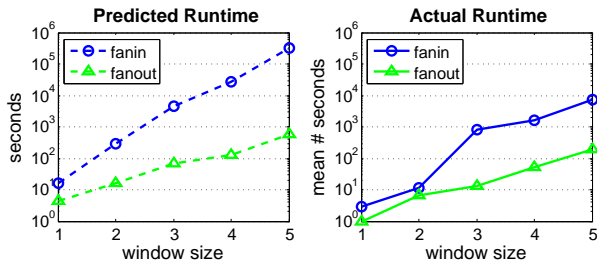


Figure 4: Predicted (left) and actual (right) OIS computation time vs. window size.

At a high level, the trend we predicted using our theoretical characterization matches the trend observed in actual data: for weakly-coupled problems that have a low degree of influence (because of their small task windows), the agent scope size has little effect on the computation time. However, as the window size increases, the size of the agent scope makes exponentially more difference. Such a prediction could not have been made, nor this trend well understood, without taking into account two different aspects of weak coupling: *degree of influence* and *agent scope size*. While our predicted runtimes appear to overestimate the actual runtimes in both cases, we do not expect the actual runtimes to precisely match predictions made using worst-case complexity bounds. For instance, Equation 4 does not account for the fact that in *fanin* problems, only a single agent is computing a best response. (In general, $\omega = 3$ topologies would require that all agents compute best responses.)

5. RELATED WORK

The first author’s dissertation [21] includes a more rigorous, though less general, treatment of the theory presented in Sections 2–3. Compared to other treatments of weak coupling in multiagent sequential decision making, the primary distinctions of our analysis are (1) its synthesis of three aspects of weak coupling into a single unified characterization, and (2) its quantification of the computational benefits of exploiting weakly-coupled structure in a *general* context of optimal Dec-POMDP planning. Each of these aspects has appeared in the literature in some shape or form. For example, the work of Guestrin *et al.* [7] on exploiting restricted state factor scope and agent scope (both of which have also been referred to under the heading *locality of interaction* [9, 15, 16]) in factored value functions, though limited in context to approximate solution computation, plays a foundational role in our analysis.

The effect of restricted agent scope on problem hardness has been previously explored in MMDPs [5] (assuming fully-jointly-observable state), and also in ND-POMDPs [9, 10, 15] (assuming transition and observation independence). In the latter, Kumar and Zilberstein [10] make an explicit connection between induced width and complexity. Oliehoek *et al.* [16] analyze the stage-by-stage dynamics of state factor scope and agent scope in factored Dec-POMDPs, treating the planning problem as a series of collaborative graphical Bayesian games. Degree of influence, is grounded in the work of Becker *et al.* [2], who identify the *coverage set* as a subset of local policies that need be considered in a joint policy search, and the works of Rathnasabapathy *et al.* [18] and Pynadath and Marsella [17], who define *behavior(al) equivalence* over candidate models of an agent’s peers. Our earlier work [22] on influence-based abstraction can be viewed as a means of partitioning the policy space into impact equivalence classes, each of which is summarized by an *influence*.

Aside from the three aspects on which our characterization concentrates, researchers have analyzed the relationships between other forms of interaction structure and problem hardness. For instance, Goldman and Zilberstein [6] characterize the complexity of various Dec-POMDP subclasses by classifying agents’ direct communication and indirect sharing of information through observation. Shen *et al.* [19] characterize complexity of optimal Dec-MDP planning according to the complexity of the minimal encoding of agents’ local policies. Allen and Zilberstein [1] develop an information-

theoretic metric, *influence gap*, that quantifies the difference in the degree to which each agent can affect world state transitions and joint rewards in a two-agent Dec-POMDP. Though the semantics of *influence* in Allen and Zilberstein’s work differ substantially from ours, their results express the same general sentiment that varying levels of impact result in varying problem hardness.

Lastly, there is a strong connection between our analysis and that of Brafman and Domshlak [3], who transform the classical planning problem into one of *constraint satisfaction*. As in our analysis of joint policy computation as constraint optimization, they incorporate a parameter ω corresponding to the induced width of the constraint graph.

6. CONCLUSIONS AND FUTURE WORK

This paper takes an important step towards gaining a better understanding of what makes some Dec-POMDP problems so much harder to solve than others. We have jointly characterized three aspects of the weakly-coupled problem structure that, when exploited, accommodate quantifiable computational gains. By studying these aspects in a unified context, we have derived new bounds on the complexity of optimal multiagent planning.

Our theoretical results complement the abundance of recent algorithmic development geared towards solving problems with structured agent interactions [2, 9, 10, 13, 14, 15, 16, 22], by providing a gauge of problem difficulty based upon the degree to which a problem is weakly-coupled. We have demonstrated that our theory can explain observations about algorithm performance, as well as predict the relative computational overhead of algorithms that exploit some or all of the elements of weakly-coupled problem structure that we have characterized. These explanations and predictions could not have been formed without considering the combination of different aspects of weakly-coupled structure.

In the process of predicting the computation time of one such algorithm, OIS [22], our empirical analysis illustrated that OIS’s past success was due, in part, to its exploitation of structure in problems with a low degree of influence. However, our analysis also exposed the need for a better understanding of the identifiable problem attributes that affect *degree of influence*, whose underlying structure is less discernible than that of *state factor scope domain size* and *agent scope size*. Moreover, in the future, we hope to find other advantageous structural aspects that would improve the predictive power of our characterization. Such a pursuit could not only expand our understanding of the performance of existing algorithms, but guide the design of better algorithms.

7. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their thoughtful comments, and Frans Oliehoek for his constructive feedback. This work was supported, in part, by NSF grants IIS-0534280 and IIS-0964512, and by AFOSR grant FA9550-07-1-0262.

8. REFERENCES

- [1] M. Allen and S. Zilberstein. Agent influence as a predictor of difficulty for decentralized problem-solving. In *AAAI*, pages 688–693, 2007.
- [2] R. Becker, S. Zilberstein, V. Lesser, and C. Goldman. Solving transition independent decentralized Markov Decision Processes. *JAIR*, 22:423–455, 2004.
- [3] R. Brafman and C. Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In *ICAPS*, pages 28–35, 2008.
- [4] R. Dechter. *Constraint Processing*. Morgan K., 2003.
- [5] D. Dolgov and E. Durfee. Graphical models in local, asymmetric multi-agent Markov decision processes. In *AAMAS*, pages 956–963, 2004.
- [6] C. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *JAIR*, 22:143–174, 2004.
- [7] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored MDPs. In *NIPS*, pages 1523–1530, 2001.
- [8] A. Guo and V. Lesser. Planning for weakly-coupled partially observable stochastic games. In *IJCAI*, 2005.
- [9] Y. Kim, R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Exploiting locality of interaction in networked distributed POMDPs. In *AAAI Spring Symp. on Distributed Planning and Scheduling*, 2006.
- [10] A. Kumar and S. Zilberstein. Constraint-based dynamic programming for decentralized pomdps with structured interactions. In *AAMAS*, pages 561–568, 2009.
- [11] J. Marecki and M. Tambe. Planning with continuous resources for agent teams. In *AAMAS*, pages 1089–1096, 2009.
- [12] N. Meuleau, M. Hauskrecht, K.-E. Kim, L. Peshkin, L. P. Kaelbling, T. Dean, and C. Boutilier. Solving very large weakly coupled Markov decision processes. In *AAAI ’98/IAAI ’98*, pages 165–172, 1998.
- [13] H. Mostafa and V. Lesser. Offline Planning For Communication By Exploiting Structured Interactions In Decentralized MDPs. In *Proceedings of IAT*, 2009.
- [14] R. Nair, M. Tambe, M. Yokoo, D. V. Pynadath, and S. Marsella. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *IJCAI*, pages 705–711, 2003.
- [15] 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.
- [16] F. A. Oliehoek, M. T. J. Spaan, S. Whiteson, and N. A. Vlassis. Exploiting locality of interaction in factored Dec-POMDPs. In *AAMAS*, pages 517–524, 2008.
- [17] D. V. Pynadath and S. Marsella. Minimal mental models. In *AAAI*, pages 1038–1044, 2007.
- [18] B. Rathnasabapathy, P. Doshi, and P. Gmytrasiewicz. Exact solutions of interactive pomdps using behavioral equivalence. In *AAMAS*, pages 1025–1032, 2006.
- [19] J. Shen, R. Becker, and V. Lesser. Agent Interaction in Distributed MDPs and its Implications on Complexity. In *AAMAS*, pages 529–536, 2006.
- [20] 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*, pages 313–320, 2009.
- [21] S. Witwicki. *Abstracting Influences for Efficient Multiagent Coordination Under Uncertainty*. PhD thesis, University of Michigan, January 2011.
- [22] S. Witwicki and E. Durfee. Influence-based policy abstraction for weakly-coupled Dec-POMDPs. In *ICAPS*, pages 185–192, 2010.

GUARDS - Game Theoretic Security Allocation on a National Scale

James Pita, Milind Tambe, Chris Kiekintveld*, Shane Cullen**, Erin Steigerwald***

University of Southern California, Los Angeles, CA 90089

*University of Texas at El Paso, El Paso, TX 79968

**APEX STORE-Technology Lead: Science and Technology Directorate,
Department of Homeland Security

***Program Manager: Transportation Security Administration

ABSTRACT

Building on research previously reported at AAMAS conferences, this paper describes an innovative application of a novel game-theoretic approach for a *national scale* security deployment. Working with the United States Transportation Security Administration (TSA), we have developed a new application called GUARDS to assist in resource allocation tasks for airport protection at over 400 United States airports. In contrast with previous efforts such as ARMOR and IRIS, which focused on one-off tailored applications and one security activity (e.g. canine patrol or checkpoints) per application, GUARDS faces three key issues: (i) reasoning about hundreds of heterogeneous security activities; (ii) reasoning over diverse potential threats; (iii) developing a system designed for hundreds of end-users. Since a national deployment precludes tailoring to specific airports, our key ideas are: (i) creating a new game-theoretic framework that allows for heterogeneous defender activities and compact modeling of a large number of threats; (ii) developing an efficient solution technique based on general purpose Stackelberg game solvers; (iii) taking a partially centralized approach for knowledge acquisition and development of the system. In doing so we develop a software scheduling assistant, GUARDS, designed to reason over two agents — the TSA and a potential adversary — and allocate the TSA's limited resources across hundreds of security activities in order to provide protection within airports.

The scheduling assistant has been delivered to the TSA and is currently under evaluation and testing for scheduling practices at an undisclosed airport. If successful, the TSA intends to incorporate the system into their unpredictable scheduling practices nationwide. In this paper we discuss the design choices and challenges encountered during the implementation of GUARDS. GUARDS represents promising potential for transitioning years of academic research into a nationally deployed system.

Categories and Subject Descriptors

J.m [Computer Applications]: MISCELLANEOUS

General Terms

Security, Design, Performance

Cite as: GUARDS - Game Theoretic Security Allocation on a National Scale, James Pita, Milind Tambe, Chris Kiekintveld, Shane Cullen and Erin Steigerwald, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems – Innovative Applications Track (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 37-44.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Keywords

Applications, Game Theory, Security, Resource Allocation

1. INTRODUCTION

The United States Transportation Security Administration (TSA) is tasked with protecting the nation's transportation systems [2]. These systems are often large in scale and require many personnel and security activities to protect them. One set of systems in particular is the over 400 airports [2]. These airports serve approximately 28,000 commercial flights per day and up to approximately 87,000 total flights [1]. To protect this large transportation network, the TSA employs approximately 48,000 Transportation Security Officers [2]. These Security Officers are responsible for implementing security activities at each individual airport in order to provide security for the transportation network.

While many people are aware of common security activities, such as individual passenger screening, this is just one of many security layers TSA personnel implement to help prevent potential threats [2]. These layers can involve hundreds of heterogeneous security activities executed by limited TSA personnel leading to a complex resource allocation challenge. Unfortunately, TSA cannot possibly run every security activity all the time and thus must decide how to appropriately allocate its resources among the layers of security to protect against a number of potential threats.

To aid the TSA in scheduling resources in a risk-based manner, we take a multi-agent game-theoretic approach. Motivated by advantages of such an approach reported at AAMAS conferences (see Section 2.2), we utilize Stackelberg games where one agent (the leader) must commit to some strategy first and a second agent (the follower) can make his decision with knowledge of this commitment. Here, the TSA acts as a defender (i.e. the leader) who has a set of targets to protect, a number of security activities to protect each target, and a limited number of resources to assign to these security activities. This approach then models a motivated attacker's ability to observe the TSA's resource allocations before choosing a potential threat to execute in an attempt to attack an airport target. The advantage of our approach is in finding the optimal mixed strategy for the TSA to commit to in order to provide them with a risk-based, randomized schedule for allocating their limited resources. From the perspective of the underlying game-theoretic model, a crucial difference of our novel approach and previous approaches is this: we allow for both heterogeneous security activities and threats whereas previous "security games" approaches reported at AAMAS [14, 15] are only able to consider homogeneous security activities and threats, leading to a new game model called "Security Circumvention Games" (SCGs).

In conjunction with TSA subject matter experts, we developed a software system, Game-theoretic Unpredictable and Randomly Deployed Security (GUARDS), that utilizes a Stackelberg framework to aid in protecting the airport transportation network. From an application perspective, the fundamental novelty in GUARDS, compared to previous applications [14, 15] of such game-theoretic approaches, is the potential national scale deployment at over 400 airports. Given that previous approaches only dealt with a single standalone location, this scale raises three new issues. The first issue is in appropriately modeling the TSA’s security challenges in order to achieve the best security policies (mixed strategy). Due to the complex nature of TSA’s security challenges, traditional models of security games [17] are no longer appropriate models. Specifically, the TSA’s domain has the following additional features beyond traditional security games: (i) heterogeneous security activities for each potential target; (ii) heterogeneous threats for each potential target; (iii) unique security activities for individual airports. The second issue is in efficiently solving the model we developed where, because we consider a national deployment, a special-purpose solver may not be appropriate. In fact, previous solution techniques [8, 9] for traditional security games are no longer directly applicable. The final issue is in knowledge acquisition for the many variables involved in TSA’s security challenges.

In consideration of national deployment for the TSA, we face two unique constraints. First, headquarters cannot do centralized planning where they create a single optimal mixed strategy (security policy) that will be applicable to all airports. Each airport is unique and thus will require its own individual security policy. Second, TSA wants to maintain a common standard of security among airports. This precludes an entirely decentralized approach where each individual airport is completely in charge of creating their security policy. Even so, due to the possibility of over 400 end-users, it is not practical to sit down with each location and tailor the system to their individual needs. This presents a challenge in acquiring the necessary domain knowledge for such a large network of airports to appropriately model their security challenge.

To address these issues, we developed both a new formal model of security games and techniques to solve this class of games. We also had to incorporate a new methodology for knowledge acquisition. To appropriately model the TSA’s security challenges we created a novel game-theoretic model, which is referred to as Security Circumvention Games (SCGs), and cast the TSA’s challenges within this model. In the creation of SCGs we provide the following contributions: (i) the ability for defenders to guard targets with more than one type of security activity (heterogeneous activities); (ii) the ability for attackers to choose threats designed to circumvent specific security activities. Given our new model, we designed an efficient solution technique in which we create a compact representation of SCGs. This allows us to avoid using a tailored Stackelberg solver and instead utilize a general purpose Stackelberg solver to compute solutions efficiently. Finally, we took a partially centralized approach to knowledge acquisition for the TSA domain. We integrated a two phase knowledge acquisition process in which we acquire common information, standards, and practices directly from TSA headquarters and then developed the GUARDS system itself to acquire the necessary information that is unique to individual airports.

These key issues present a novel and exciting problem in transitioning years of research from the AAMAS conference to a highly complex domain [9, 12, 14, 15, 17]. GUARDS is currently under evaluation by the TSA with the goal of incorporating its scheduling practices into their unpredictable security programs across airports nationwide.

2. BACKGROUND

Game theory is well known to be a useful foundation in multi-agent systems to reason about multiple agents each pursuing their own interests [7]. Game-theoretic approaches, specifically based on Stackelberg games, have recently become popular as approaches to address security problems (e.g. assigning checkpoints, air marshals, or canine patrols). These approaches reason about two agents pursuing opposing interests (i.e. a security force and an adversary) in an attempt to optimize the security force’s goals. Specifically, they model the commitment a security force must make in providing security and the attacker’s capability of observing this commitment before attacking. The objective is to find the optimal mixed strategy to commit to given that an attacker will optimize his reward after observing this strategy. At this point we will describe how security games, as defined in [17], fit into the Stackelberg paradigm. In Section 3.1 we will define SCGs to account for the challenges that the TSA faces.

2.1 Security Games

In a security game there are two agents – the defender (security force) and an attacker – who act as the leader and the follower in a Stackelberg game. There are also a set of targets, which the defender is trying to protect. Each of these targets has a unique reward and penalty to both the defender and attacker. Thus, some targets may be more valuable to the defender than others. To protect these targets the defender has a number, K , of resources at her disposal. There is a single security activity being considered and these resources can be allocated to execute this activity on any target. Once a resource is allocated to a target it is marked as covered, otherwise it is marked as uncovered. If the attacker attacks an uncovered target he gets his reward and the defender her corresponding penalty else vice versa. The defender’s goal is to maximize her reward given that the attacker will attack with knowledge of the defensive strategy the defender has chosen. In most cases, the optimal strategy for the defender is a randomized strategy in which she chooses a mixed strategy over all her possible resource assignments.

There exist a number of algorithms and techniques for solving security games [6, 8, 9, 12]. DOBSS, a mixed-integer linear program, and the Multiple Linear Programs methods are the most general and are capable of solving any Stackelberg game optimally [6, 12]. The other algorithms are tailored to security games specifically and are much faster in practice for these games.

2.2 Assistants for Security Games

A number of tools have been designed to assist in security problems that fall under the security game paradigm. ARMOR and IRIS are two such tools which take a game-theoretic approach for scheduling checkpoints and canine patrols (ARMOR) and Federal Air Marshals (IRIS). In fact, ARMOR and IRIS have been deployed to aid with security operations for the Los Angeles World Airport Police at Los Angeles International airport and for the Federal Air Marshals Service respectively [14, 15]. These systems offer two sets of advantages. The first set of advantages deal with solution quality: (i) they provide an optimal mixed strategy for the single security activity they consider such as assigning checkpoints or air marshals; (ii) the randomized solutions produced both avoid deterministic strategies that are easily exploitable and remove the human element in randomization since humans are well known to be poor randomizers [16]; (iii) they reason over difficult problems that are often impossible for humans to reason over optimally. These advantages are useful for any tool being utilized in the field to help with randomized resource allocation in security problems and we incorporate them into GUARDS as well.

The second set of advantages are specific to the problems they address: (i) they develop unique and useful preference elicitation systems and knowledge acquisition techniques for the specific problem they address; (ii) based on practical requirements, they apply state of the art algorithms tailored to solving the Stackelberg games they consider efficiently. Unfortunately, previous methods are too specific to the standalone location they consider and thus cannot directly be applied in GUARDS; indeed GUARDS requires us to address a novel set of challenges described in the next section.

3. NATIONAL DEPLOYMENT CHALLENGES

We now describe in detail the three major issues in potentially deploying game-theoretic randomization for airport security on a national scale, including modeling, computational, and knowledge acquisition challenges and our solutions to them.

3.1 Modeling the TSA Resource Allocation Challenges

While we are motivated by an existing model of security games [17], there are three critical aspects of the new TSA domain that raise new challenges. First, the defender now reasons over heterogeneous security activities for each potential area within an airport¹. For example, airports have ticketing areas, waiting areas, and cargo holding areas. Within each of these areas, TSA has a number of security activities to choose from such as perimeter patrols, screening cargo, screening employees and many others. Second, given the multiple possible security activities, the defender may allocate more than one resource per area (i.e. areas are no longer covered or uncovered). Finally, the defender now considers an adversary who can execute heterogeneous attacks on an area. The TSA must reason about a large number of potential threats in each area such as chemical weapons, active shooters, and bombs. The key challenge is then how to allocate limited TSA security resources to specific activities in particular areas, taking into account an attacker’s response.

To address this challenge it is necessary to create a more expressive model than outlined in security games; one that is able to reason over the numerous areas, security activities, and threats within an individual airport. We refer to this new class of security games as Security Circumvention Games (SCGs). SCGs are more expressive than traditional security games and thus can represent both traditional security games and the games we considered for the TSA. In SCGs, the TSA must choose some combination of security activities to execute within each area and the attacker must reason over both which area to attack and which method of attack to execute based on the defender’s strategy. At this time we elaborate on the defender’s and attacker’s possible strategies.

3.1.1 Defender Strategies

We denote the defender by Θ , and the set of defender’s pure strategies by $\sigma_\Theta \in \Sigma_\Theta$. The TSA is able to execute a variety of security activities, which we denote by $S = \{s_1, \dots, s_m\}$. Each security activity has two components. The first is the type of activity it represents, and the second is the area where the activity is performed. We denote the set of areas by $A = \{a_1, \dots, a_n\}$.

The defender has K resources available and thus can run any K security activities. The TSA’s task is to consider how to allocate these resources among security activities in order to provide the optimal protection to their potential areas. An assignment of

¹Due to the nature of the TSA’s security challenge, we will refer to targets in the TSA’s domain as areas henceforth.

K resources to K security activities represents a single strategy $\sigma_\Theta \in \Sigma_\Theta$. For example, if there are three security activities, $S = \{s_1, s_2, s_3\}$ and two resources available, one possible pure strategy for the defender is to assign these two resources to s_1 and s_3 . Given that the number of possible combinations of K security activities at an airport can be on the order of 10^{13} or greater for the TSA, we develop a compact representation of the possible strategies that we present in Section 3.2. The defender’s mixed strategies $\delta_\Theta \in \Delta_\Theta$ are the possible probability distributions over Σ_Θ . Similar to previous work, a mixed strategy (randomized solution) is typically the optimal strategy.

3.1.2 Attacker Actions

Defending a target against terrorist attacks is complicated by the diversity of the potential threats. For example, an attacker may try to use a vehicle borne explosive device, an active shooter, a suitcase bomb, and many others in any given area. Not all methods of attack would make sense in all areas. For example, using a vehicle borne explosive device in the checked baggage screening area in some airport configurations would not be a viable method of attack. We denote the attacker by Ψ , and the set of pure strategies for the attacker is given by $\sigma_\Psi \in \Sigma_\Psi$. Each pure strategy for the attacker corresponds to selecting a single area $a_i \in A$ to attack, and a specific mode of attack. However, given that each airport considers its own potential threats, enumerating all threats for each individual airport through the software may not be practical. To handle the national deployment challenge we face and avoid this difficulty, we developed a novel way to represent threats for TSA’s domain that we describe in Section 3.2.1.

3.2 Compact Representation for Efficiency

While we have developed a model that appropriately captures the TSA’s security challenge, one issue with this model is that both the attacker and defender strategy spaces grow combinatorially as the number of defender security activities increases. Also, listing such a large number of potential threats would lead to extreme memory and runtime inefficiencies. Furthermore, existing solution techniques that have been developed for security games [8, 9] are not directly applicable to Security Circumvention Games (SCGs).

With this in mind, we looked at an alternate approach to finding optimal solutions efficiently. Specifically, we looked at representing threats in a more intelligent manner and creating a compact representation for the defender strategy space. By utilizing both of these techniques, we achieved large reductions in run-time. We utilized a general Stackelberg solver known as DOBSS [12] to solve our compact representation and avoided creating a tailored algorithm for each specific airport. At this time we will explain both how we model threats and how we achieve a compact representation of the defender’s full strategy space.

3.2.1 Threat Modeling for TSA

While it is important that we reason over all the security activities that are available to an individual airport, enumerating all of the large number of potential threats they face can lead to severe memory and runtime inefficiencies. Thus, the problem we face is how to model attack methods in a way that limits the number of threats GUARDS needs to reason over, but appropriately captures both an attacker’s capabilities and his goals. In particular, we automatically generate attack methods for the adversary that capture two key goals: (i) an attacker wants to avoid the security activities that are in place; (ii) an attacker wants to cause maximal damage with minimum cost.

In order to achieve these goals an intelligent adversary will ob-

serve security over time and design his attack method based on his observations. The attacker’s plan will be designed to avoid security activities that he believes will be in place. We will refer to this as circumventing security activities. For example, imagine there is a single area with three security activities such as passenger screening, luggage screening, and perimeter patrol. In this example, TSA only has one resource available and thus can only execute one of these activities at a time. While passenger screening may have the highest probability of success, if TSA never screens luggage or patrols the perimeter, the adversary can choose an attack path that avoids passenger screening such as utilizing a suitcase bomb or an attack from the perimeter.

On the defender side, we know that dedicating more resources to security activities in an area increases the security afforded to that area. However, even with more resources, we want to avoid being predictable since attackers can exploit this predictability; avoiding the security activities they know will be in place. Thus, we needed to represent threats in a way that accounts for the attacker’s ability to observe security in advance and avoid specific security activities, but still represents the benefit of dedicating more resources.

A naïve approach is to represent only a single threat per area and decrease the likelihood of success for that threat as more security activities are put in place. This captures the increase in security for additional security activities, however, it does not account for the attacker’s ability to circumvent security activities. With this method you would simply choose security activities in the order of their relative success making it predictable and exploitable.

The alternative that we chose is to create a list of potential threats that circumvent different combinations of specific security activities. By basing threats on circumventing particular combinations of security activities, we avoid the issue of enumerating all the possible potential threats. Instead the threats are automatically created based on the security activities in an area. However, we also incorporate a cost to the attacker for circumventing more activities to capture the idea of causing maximal damage at minimal cost. Each individual activity has a specific circumvention cost associated with it and more activities circumvented leads to a higher circumvention cost. This cost reflects the additional difficulty of executing an attack against increased security. This difficulty could be due to requiring additional resources, time and other factors for executing an attack. Since attackers can now actively circumvent specific security activities, randomization becomes a key factor in the solutions that are produced because any deterministic strategies can be circumvented.

3.2.2 Compact Representation

We introduce a compact representation that exploits similarities in defender security activities to reduce the number of strategies that must be enumerated and considered when finding an optimal solution to SCGs. First, we identify security activities that provide coverage to the same areas, and have the same circumvention costs (i.e. have identical properties). Let $\gamma_i \in \Gamma$ represent the sets of security activities that can be grouped together because they have identical properties. Now, instead of reasoning over individual security activities, we reason about groups of identical security activities $\gamma_i \in \Gamma$. A strategy $\sigma_\Theta \in \Sigma_\Theta$ is represented by the number of resources assigned to each set of identical security activities γ_i .

To illustrate this new representation, we provide a concrete example of the full representation versus the compact representation in Tables 1 and 2. In this example there are 4 security activities and 2 resources. Here, s_1 and s_2 have identical circumvention costs and affect a_1 while s_3 and s_4 have identical circumvention costs and affect a_2 . Table 1 presents the full representation with corre-

sponding payoffs and Table 2 represents the compact form of the same where γ_1 represents the group s_1 and s_2 and γ_2 represents the group s_3 and s_4 . In both tables, each row represents a single pure strategy for the defender and each column the same for the attacker. Notice in Table 1 each strategy $\sigma_\Theta \in \Sigma_\Theta$ is represented by the exact security activities being executed while in Table 2 it is only which set $\gamma_i \in \Gamma$ each resource has been allocated to.

The key to the compact representation is that each of the security activities from a set $\gamma_i \in \Gamma$ will have the same effect on the payoffs. Therefore, it is optimal for the defender to distribute probability uniformly at random across all security activities within a set γ_i , so that all security activities are chosen with equal probability in the solution. Given that the defender strategy uniformly distributes resources among all security activities $s_j \in \gamma_i$ we also know that it does not matter which specific security activities the attacker chooses to circumvent from the set γ_i . For any given number of security activities circumvented, the expected payoff to the attacker is identical regardless of which specific activities within the set are chosen. This is because we are selecting security activities uniformly at random within the set γ_i . Therefore, we can use a similar compact representation for the attacker strategy space as for the defender, reasoning only over the aggregate number of security activities of each type rather than specific security activities.

Given this, we only need to know how many security activities are selected from each set in order to compute the expected payoffs for each player in the compact representation. For example, examining the second row and second column of Table 2 we see that the reward to the defender is -2 and the reward to the attacker is 0. In this case, the defender strategy is to assign 1 resource to activities in γ_1 and 1 resource to activities in γ_2 . Given that she is uniformly distributing these resources, it follows that she will execute s_1 half of the time and s_2 the other half. On the attacker side, we know that the attacker is circumventing one security activity from the set γ_1 . If he circumvents either s_1 or s_2 he will only succeed half of the time. Thus, half of the time the defender receives 4 and the other half -8 for an expectation of $-2 (4 * .5 + (-8) * .5)$. We compute the attacker’s reward in the same manner.

	$a_1 : \emptyset$	$a_1 : s_1$	$a_1 : s_2$	$a_2 : \emptyset$	$a_2 : s_3$	$a_2 : s_4$
s_1, s_2	2, -1	4, -3	4, -3	-20, 10	-17, 7	-17, 7
s_1, s_3	2, -1	-8, 3	4, -3	5, -5	-17, 7	8, -8
s_1, s_4	2, -1	-8, 3	4, -3	5, -5	8, -8	-17, 7
s_2, s_3	2, -1	4, -3	-8, 3	5, -5	-17, 7	8, -8
s_2, s_4	2, -1	4, -3	-8, 3	5, -5	8, -8	-17, 7
s_3, s_4	-10, 5	-8, 3	-8, 3	5, -5	8, -8	8, -8

Table 1: Example payoffs for sample game.

	$a_1 : \emptyset$	$a_1 : \gamma_1$	$a_2 : \emptyset$	$a_2 : \gamma_2$
γ_1, γ_1	2, -1	4, -3	-20, 10	-17, 7
γ_1, γ_2	2, -1	-2, 0	5, -5	-4.5, -5
γ_2, γ_2	-10, 5	-8, 3	5, -5	8, -8

Table 2: Example compact version of sample game.

Given this compact representation for both the defender and attacker, we can compute an optimal mixed strategy of assigning resources over Γ . Once we have this mixed strategy, we will need to determine an actual strategy for the TSA to execute by sampling one of the possible strategies from the mixed strategy we have determined for our compact representation (e.g. one sample may be $\gamma_2 \gamma_2$). Once sampled, we will know exactly how many resources are available to each set $\gamma_i \in \Gamma$. Given this resource assignment,

we can then sample security activities by selecting k uniformly at random where k is the number of resources assigned to $\gamma_i \in \Gamma$. This specific set of security activities for each area under the current resource assignment is a full strategy for the TSA to execute.

3.3 Knowledge Acquisition

One of the most difficult issues we faced from a potential national deployment perspective was in acquiring the appropriate knowledge for the security challenge being considered. In the past, tools such as ARMOR and IRIS [14, 15] have been developed to be used for a single security activity in a standalone location. That approach gave the advantage of being able to sit down with domain experts who will be using the system and develop a knowledge acquisition process for the specific domain at hand. Unfortunately, with hundreds of airports to consider, it is not possible to sit down at each location and acquire the exact needs for each of them. To overcome this obstacle, in close collaboration with TSA headquarters we developed a two phase knowledge acquisition process.

In phase one, we take an approach similar to previous centralized approaches. In particular, we met with domain experts to acquire knowledge that is common among all airports. This included area definitions, defining security activities, and determining resource capabilities among others. In collaboration with headquarters, we then decided how individual airports can customize these components in their individual games while maintaining standards set forth by headquarters (as discussed below). Additionally, we collaborated with headquarters to limit the amount of customization inputs so users at individual airports are not overwhelmed – a key to organizational acceptance as discussed in Section 6.

In phase two of our knowledge acquisition, we took a decentralized approach where it is the responsibility of individual airports to input customized information. For this phase we could not create a rigid system that was designed with one specific game instance in mind for a single airport. Instead, we rely on SCGs and developed a system in collaboration with headquarters that allows individual airports to manipulate specific components within this framework to create unique game instances. These inputs are designed to ensure that individual airports maintain standards set forth by headquarters in phase one. For example, individual airports are responsible for determining the unique reward and penalty associated with each area for the defender and attacker given a successful or unsuccessful attack. However, TSA headquarters requires a standardized method for determining these values to ensure that resources are being appropriately distributed. To this end, we designed an input module within GUARDS to reflect a risk evaluation process developed by the TSA where a series of quantifiable questions are answered for each area by individual airports. These questions include such things as the number of fatalities that may result from an attack in an area, whether the area has access control, and others. The answers to these questions are then combined in a mathematical formula to decide the values for a particular area for both the defender and attacker². This input process ensures that airports are appropriately valuing the areas they protect within an airport according to headquarters guidelines. In general, using the customizable input airports generate, we can then create the unique game instance for that particular airport.

Our two phase knowledge acquisition process follows a partially centralized approach and provides the following advantages: (i) it allows domain experts from TSA headquarters to assure that the system meets the required needs of the challenge being considered;

²Previous work [14, 15, 17] has shown that security problems are not necessarily zero-sum for a number of potential reasons. In GUARDS, for similar reasons, games are not necessarily zero-sum.

(ii) it focuses on creating customizable inputs instead of a system tailored to a highly specific problem instance; (iii) it allows TSA headquarters control while still enabling individual airports to customize the system to meet their individual needs. For the third advantage, there is an important trade-off between system customization and standardization among airports. Determining this trade-off is an important part of the first phase in this two phase knowledge acquisition process.

4. SYSTEM ARCHITECTURE

The GUARDS system consists of three modules. First, there is an input module designed to acquire the necessary information for one unique instance of the complex security game we consider. Second, there is a back-end module that is designed to both create and solve the unique game instance based on the inputs. Finally, there is a display/output module that presents a sample schedule to TSA officials based on the optimal solution. We now describe each individual module and its operations in TSA’s airport domain.

Input Module: The input module is composed of three classes of inputs that are required by the system in order to generate a representative Stackelberg game and create an optimal allocation of resources. All inputs are quantifiable and tangible so that headquarters is able to maintain standards and guidelines on the way security policies are created. The first input is the area data. First, airports must input each of their potential areas. Second, for each area the airport must go through the risk evaluation process (see Section 3.3) which involves answering a series of quantifiable questions. The second set of input is the security activities data. For each area the airport must list all of the security activities that are available to execute in that area. While there is a standard list of activities airports can select from, they are also able to input new security activities that may be unique to that airport. The third input is the resource data. This includes the number of days to create a schedule for and the number of resources available each day.

Back-end Module: The back-end module has three primary components. These are generating the game, solving the game, and returning one sample schedule for TSA’s use. First, based on the inputs from the input module, there is a component that creates a compact representation of the specific game instance the system is considering. This game instance is based on the compact form of the model we presented in Section 3.1. Second, we compute the solution to the Stackelberg game model using DOBSS, a general Stackelberg solver [12]. This produces a solution, which is a mixed strategy over the possible action space as defined in Section 3.2. Finally, using the optimal mixed strategy we sample one possible resource assignment that can be implemented by TSA.

Display/Output Module: The actual resource assignment selected is presented to the user via the display/output module. The schedule created is shown in the interface first as a summary of the number of resources assigned to each area similar to the mockup in Figure 1³. Once the schedule is created, TSA personnel can proceed to a more in depth report of the schedule. This report lists each of the specific security activities that were chosen for each location along with specific details of these security activities. After reviewing the report, TSA personnel can also choose to examine the distribution of resources over areas that the optimal mixed strategy provides as in Figure 2.

5. EVALUATION

³We are unable to show actual screen shots from our system due to security concerns. For the remainder of this paper we will show only basic visual representations of what GUARDS displays.

Locations	8/15	8/16	8/17	8/18	8/19	8/20	8/21
A1	1	3	1	1	3	0	0
A2	0	2	1	0	0	0	0
A3	2	2	0	1	0	3	1
A4	1	0	3	2	0	1	2
...
Total	10	10	10	10	10	10	10

Figure 1: Summary of sample schedule

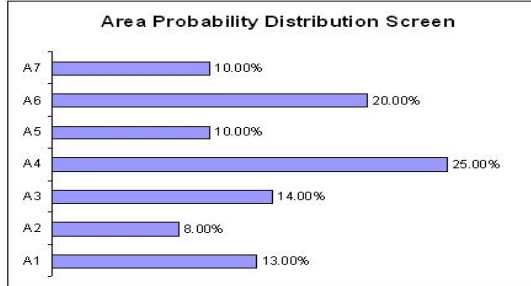


Figure 2: Summary of probability distribution over areas

When evaluating a system like GUARDS there are two important issues that are raised. The first issue is with scalability and run-times. To be useful in practice, the system needs to be able to solve real world challenges. The second issue is evaluating the value of the security policies generated against alternative approaches. In the following sections we present each of these evaluations.

5.1 Run-time Analysis

We present simulation results focusing on the computational efficiency of our compact method versus the full representation. All experiments are run on a system with an Intel 2 GHz processor and 1 GB of RAM. We used a publicly available linear programming package called GLPK to solve optimization problems as specified in the original DOBSS procedure. For the compact version we use a slightly altered version of DOBSS that is designed specifically for efficiency in the compact representation. The solver was allowed to use up to 700 MB of memory during the solution process. For larger game instances, solving the problem with the full representation runs out of memory and solutions cannot be found. In the results presented below we exclude results for cases where the full representation was not able to produce a result using the allotted memory. We also note that in all experiments both the solution found by the full representation and the solution found by the compact representation are optimal.

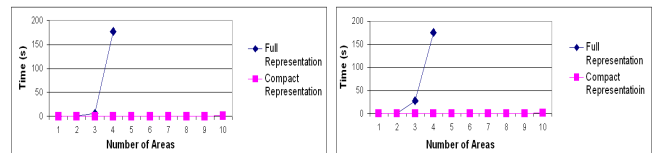
To test the solution methods we generated random game instances by randomly selecting payoff values from 1 to 50 and circumvention costs from 1 to 5 for each area. For each experiment we generated 20 random game instances and averaged the results (there is little variance in the run-times for different problem instances). We considered three different scenarios. The first scenario presents results for the case where there is an increasing number of areas, and each area has exactly 3 security activities associated with it. There are 5 resources available for the defender, and each security activity has identical properties (i.e. no security activity has a higher cost for circumvention or higher probability of success) for the area it is associated with. Given the M possible areas, for the full representation there are $\binom{3 \cdot M}{5}$ possible defender pure strategies and $8 \cdot M$ possible attacker pure strategies. Thus, in the 10 area case there are 142,506 defender pure strategies and 80 attacker

pure strategies. Examining Figure 3 (a), we show the improvement in run-time of our compact representation over the full representation. For more than 4 areas, the full representation failed to achieve a solution within the memory bounds. For 4 areas, the compact representation runs much faster than the full representation, with a run-time of less than 1 second versus the 177 seconds required by the full representation. In fact, for 10 areas, the compact representation has an average run-time of approximately 1 second, which is still much faster than the full representation for only 4 areas. Even if the number of security activities associated with each area is a relatively small constant our compact representation provides substantial benefits. As the number of similar security activities associated with an area increases, this advantage grows.

In our second scenario, we considered a situation where security activities are distributed randomly across possible areas. The total number of security activities is set similarly to the previous experiment, in that that the total number of security activities is three times the number of areas. However, we randomly assigned security activities to areas (with each area having at least one security activity) so the number is no longer uniform across areas. Once again the defender has 5 resources available and security activities have identical properties within an area. It follows that in the full representation, the number of defender pure strategies and attacker pure strategies are identical to the previous scenario. However, the number of strategies in the compact representation for both the defender and attacker may vary. Looking at Figure 3 (b), we see similar benefits for the compact representation in this case as in the previous experiment with a uniform distribution of activities.

In the final scenario, we considered a situation in which there are 10 areas to protect, each area has 3 identical security activities, and we increased the number of resources available to distribute between these areas. Thus, in the full representation, assuming there are K resources available, the defender has $\binom{30}{K}$ possible pure strategies and the attacker has 80 possible pure strategies. In Figure 4, we increase the number of resources available along the x-axis and show the time to compute a solution in seconds on the y-axis. The full representation is unable to compute a solution for more than 4 resources under these conditions within the allotted memory. On the other hand, the compact representation is able to arrive at a solution for 10 available resources in less than 30 seconds.

These results show the benefits of our compact representation in terms of efficiency. We obtained further efficiency gains by caching results: specifically, the inputs into the game do not change on a daily basis. Thus, we can cache the resulting mixed strategy, and present results from sampling this mixed strategy, as long as the program users have not changed the inputs. When they do change inputs, we resolve the game using our compact representation.



(a) Three activities per area (b) Random activities per area

Figure 3: X-axis: Areas, Y-axis: Run-time

5.2 Security Policy Analysis

For this analysis we examined the security policies generated by our game representation against two other possible solution strategies. The first strategy is a solution concept where resources are distributed uniformly among areas (uniformly random), an approach

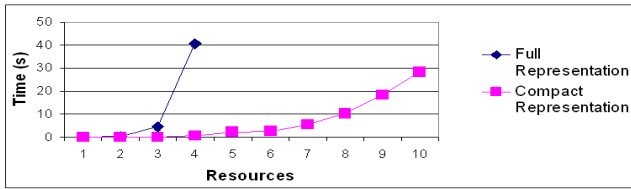


Figure 4: Run-time: Increasing resources for 10 areas with 3 security activities per area

sometimes used in lieu of a game-theoretic approach. The second strategy uses our new representation, however, it does not allow attackers to circumvent security activities (SCGs without circumvention). That is, we allow the attacker only a single attack strategy per area and simply reduce the value of that strategy as the number of security activities increases. This is a simplified model of an attacker as mentioned in Section 3.2.1. Finally, we included our new representation and allow an intelligent attacker to circumvent specific security activities when planning his mode of attack (SCGs).

We generated 20 random game instances with 10 areas and 3 security activities per area. In each game instance the payoff value of each area for both the defender and attacker are randomly selected from 1 to 50 and the circumvention costs are similarly selected from 1 to 5. We then calculated the optimal solution under the current solution strategy (i.e. uniformly random, SCGs without circumvention, and SCGs). After finding the optimal solution, we determined the expected reward for each solution given the assumptions made in SCGs (i.e. attackers are allowed to circumvent specific security activities when planning their attack). For each game instance, we computed the optimal solution varying the number of resources available from 1 to 10 as seen on the x-axis of Figure 5. On the y-axis, we present the average expected reward obtained by each solution strategy across all 20 game instances. In Figure 5 we see that the uniform policy is outperformed by both game-theoretic approaches with the approach accounting for circumvention strategies performing the best. In fact, an approach that accounts for circumvention strategies is the only one that was able to obtain a positive reward for the defender in the 20 randomly generated game instances and in the 10 resource case obtains a 200% improvement in reward over any other strategy. This shows the benefits of reasoning about an intelligent attacker who will research and exploit deterministic security activities.

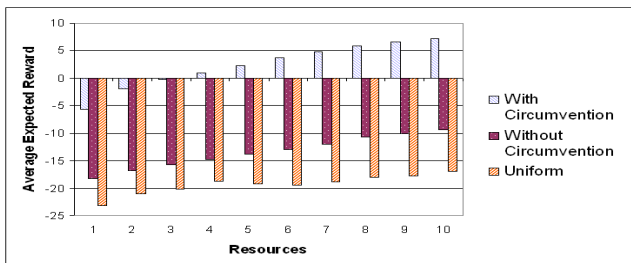


Figure 5: Policy Analysis: Increasing resources for 10 areas with 3 security activities per area

6. LESSONS IN TRANSITIONING RESEARCH INTO PRACTICE

GUARDS is the result of a unique collaboration where university researchers worked directly with a security agency for the purpose

of creating a useful product to potentially deploy outcomes of research on a national scale. This collaboration to transition research to such a large-scale deployment has presented valuable lessons. This section outlines the three areas of insights we have gained in the process: (i) acceptance of GUARDS at headquarters; (ii) acceptance of GUARDS by a variety of end-users at numerous airports; (iii) obtaining correct input from users. Some of these insights are contrary to accepted wisdom in the research community.

In a large organization like the TSA, it is important that they are able to provide quality guarantees. A key implication is that a system such as GUARDS must be very clear-cut in terms of its assumptions and its solution quality guarantees based on these assumptions. Researchers often assume that speedy heuristic solutions that are on average high quality may be adequate “in the field”, but we have learned in contrast that when dealing with security agencies it is important that we provide guarantees on this solution quality. More importantly, these guarantees may even be required to be optimal (i.e. even if we can guarantee solutions within some bound of the optimal solution it may not be enough). Without guarantees, the TSA may be unable to justify the use of any particular security strategy. In accordance with this requirement, we use a solver known as DOBSS, which provides game-theoretic optimal solutions in Stackelberg games.

With respect to acceptance of GUARDS at individual airports, one major lesson learned is bridging the culture gap in academic research and real-world operations. Indeed, what researchers may consider small uninteresting issues may nullify all their major research advances. For example, in an initial version of GUARDS, we displayed the final probabilities of our mixed strategies, but truncated the presentation of real numbers (i.e. truncating all decimal values). Unfortunately, this single display issue turned out to be a major headache for users who assumed incorrectness on part of GUARDS when the distribution of resources appeared to be less than 100%. Specifically, instead of considering the truncation of all real values, users might assume that some resources were not being utilized. A second major lesson learned is the continued need for efficiency of game-theoretic algorithms. While significant research has gone into speeding up these algorithms, we are still not able to get off-the shelf algorithms and deploy; GUARDS required the use of new compact representations. We have outlined our key advances in this regard in Section 5.1; including the need for caching.

A third lesson learned in user acceptance is careful design of the user interface so as to reduce the amount of user workload to provide inputs: this must be kept at a manageable level. For instance, if users are required to directly enter values into the generated game matrix it can require thousands of inputs. Instead, it is important to provide a user-friendly method of conveying the necessary information. We used a simple interface where users are only required to input the base information that is then used to generate the larger game matrix. By base information, we mean such things as the areas and security activities. This is information that they have direct access to and can easily be input by the individual airports.

Finally, in any collaboration, it is important that researchers are able to obtain the appropriate input from their collaborators. This includes understanding what information is available versus what is not and accounting for this in modeling of the problem. For the available information, often end-users will not understand the techniques being applied and thus are prone to providing vague or incorrect information. For example, when asking a security agency such as the TSA to provide a utility for an attacker and for themselves as a defender on a successful attack, they may always say that it is very bad for themselves and very good for the attacker. Specifically, if there are 5 areas and they provide a utility for each

on a 10 point scale, they may always claim that it is -10 for the defender and 10 for the attacker. In practice, this feedback may not be useful because attacks on different areas may actually have very different impact in terms of economic damage, casualties, and many other factors. To aid in preventing this scenario, it is important to convey the impact that inputs will have on outputs; aiding their understanding of how their inputs will affect the results.

7. RELATED WORK AND SUMMARY

To the best of our knowledge, this paper presents the first-ever effort to transition any research reported at AAMAS conferences to an application designed for potential national scale deployment to hundreds of locations. This contrasts with previous efforts, including efforts that focus on application of game-theoretic approaches such as ARMOR and IRIS [14, 15], as detailed earlier in the paper. It also contrasts alternative models based on Markov Decision Processes (MDPs), queuing theory, or game theoretic approaches that would enumerate all possible defender actions and attacker threats [11, 13]. To accomplish this transition, we outlined novel contributions to game modeling and compact representations of games, because of the scale-up in defender and attacker strategies. This research complements other solution techniques for Stackelberg games [5, 10], which have traditionally not focused on such a scale-up. Our work also complements research actually applied to randomize patrolling strategies in robot patrol [3, 4], given our emphasis on modeling adversaries in a game-theoretic setting.

TSA is charged with protecting over 400 airports in the US. The key challenge is how to intelligently deploy limited security resources to unpredictable security activities within the airport in a risk-based manner to provide the maximum possible protection. These decisions may be made on a daily basis, based on the local information available at each airport.

This paper describes a scheduling assistant for TSA, GUARDS, which takes a game-theoretic approach to this resource allocation task. In creating GUARDS, we address three key issues that arise from a potential national deployment case. These issues are: (i) knowledge acquisition for hundreds of end-users under one organization; (ii) appropriately modeling TSA's security challenge to achieve the best security policies; (iii) efficiently finding solutions to the problem we consider. We addressed the first challenge by using a two phase knowledge acquisition process in which we acquire common information, standards, and practices directly from TSA headquarters. We then constructed the GUARDS system itself to reflect a risk evaluation process designed by TSA to acquire the necessary information that is unique to individual airports. To address the second challenge we developed a novel game-theoretic model, which we refer to as Security Circumvention Games (SCGs), and cast TSA's security challenge within this model. In creating this model we provided the following contributions: (i) the ability for defenders to guard targets with more than one type of security activity (heterogeneous activities); (ii) the ability for attackers to choose threats designed to mitigate specific security activities. Finally, we designed an efficient solution technique for reasoning over our new game model where we rely on creating a compact representation of each game instance and solving it using a general purpose Stackelberg solver. This is in contrast to tailored algorithms of the past that are designed for specific problem instances for standalone locations. To conclude, we present results demonstrating the benefits of our contributions along with lessons learned in creating GUARDS. The scheduling assistant has been delivered to the TSA and is currently under evaluation and testing for unpredictable scheduling practices at an undisclosed airport.

8. ACKNOWLEDGMENTS

The development of GUARDS has only been possible due to the exceptional collaboration with the Transportation Security Administration. This research was supported by the United States Department of Homeland Security through the Center for Risk and Economic Analysis of Terrorism Events (CREATE) under grant number 2007-ST-061-000001. However, any opinions, findings, and conclusions or recommendations in this document are those of the authors and do not necessarily reflect views of the United States Department of Homeland Security.

9. REFERENCES

- [1] Air traffic control: By the numbers. In <http://www.natca.org/mediacenter/bythenumbers.msp#1>.
- [2] TSA | Transportation Security Administration | U.S. Department of Homeland Security. In <http://www.tsa.gov/>.
- [3] N. Agmon. On events in multi-robot patrol in adversarial environments. In *AAMAS*, 2010.
- [4] N. Agmon, S. Kraus, G. Kaminka, and V. Sadov. Adversarial uncertainty in multi-robot patrol. In *IJCAI*, 2009.
- [5] N. Basilico, N. Gatti, T. Rossi, S. Ceppi, and F. Amigoni. Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings. In *IAT*, 2009.
- [6] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *EC*, 2006.
- [7] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.
- [8] M. Jain, E. Kardes, C. Kiekintveld, M. Tambe, and F. Ordóñez. Security games with arbitrary schedules: A branch and price approach. In *AAAI*, 2010.
- [9] C. Kiekintveld, M. Jain, J. Tsai, J. Pita, M. Tambe, and F. Ordóñez. Computing optimal randomized resource allocations for massive security games. In *AAMAS*, 2009.
- [10] D. Korzhyk, V. Conitzer, and R. Parr. Complexity of computing optimal Stackelberg strategies in security resource allocation games. In *AAAI*, 2010.
- [11] R. C. Larson. A hypercube queueing model for facility location and redistricting in urban emergency services. *Computers and OR*, 1(1):67–95, 1974.
- [12] P. Paruchuri, J. Marecki, J. Pearce, M. Tambe, F. Ordóñez, and S. Kraus. Playing games for security: An efficient exact algorithm for solving Bayesian Stackelberg games. In *AAMAS*, 2008.
- [13] P. Paruchuri, M. Tambe, F. Ordóñez, and S. Kraus. Security in multiagent systems by policy randomization. In *AAMAS*, 2006.
- [14] J. Pita, M. Jain, J. Marecki, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Deployed ARMOR protection: The application of a game theoretic model for security at the Los Angeles International Airport. In *AAMAS*, 2008.
- [15] J. Tsai, S. Rathi, C. Kiekintveld, F. Ordóñez, and M. Tambe. IRIS - a tool for strategic security allocation in transportation networks. In *AAMAS*, 2009.
- [16] W. A. Wagenaar. Generation of random sequences by human subjects: A critical survey of literature. 1972.
- [17] Z. Yin, D. Korzhyk, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in security games: Interchangeability, equivalence, and uniqueness. In *AAMAS*, 2010.

Best Papers Session II

On the Outcomes of Multiparty Persuasion

Elise Bonzon
LIPADE, Université Paris Descartes
France
elise.bonzon@parisdescartes.fr

Nicolas Maudet
LAMSADE, Université Paris Dauphine
France
maudet@lamsade.dauphine.fr

ABSTRACT

In recent years, several bilateral protocols regulating the exchange of arguments between agents have been proposed. When dealing with persuasion, the objective is to arbitrate among conflicting viewpoints. Often, these debates are not entirely predetermined from the initial situation, which means that agents have a chance to influence the outcome in a way that fits their individual preferences. This paper introduces a simple and intuitive protocol for multiparty argumentation, in which several (more than two) agents are equipped with argumentation systems. We further assume that they focus on a (unique) argument (or issue) —thus making the debate two-sided— but do not coordinate. We study what outcomes can (or will) be reached if agents follow this protocol. We investigate in particular under which conditions the debate is pre-determined or not, and whether the outcome coincides with the result obtained by merging the argumentation systems.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent systems

General Terms

Theory

Keywords

Argumentation, persuasion protocols, multiagent systems

1. INTRODUCTION

Protocols for persuasion [11] regulate the exchange of arguments to arbitrate among conflicting viewpoints. Depending on the underlying objective, such protocols can be more or less flexible. When conceived as *argument games* (or *disputes*) between a proponent and an opponent, proof theoretical counterparts of argumentation semantics must leave no room for uncertainty in the result. On the other hand, when the ambition is to regulate some interaction between different agents, it is often desirable that the outcome of the dialogue is not entirely predetermined from the initial

situation [7, 8]. This means that agents have a chance to influence the outcome of the game depending on how they play.

Recently, different properties of these protocols have been studied with the help of game-theoretical concepts (see [13] for a survey). This paper follows this line of work, and develops an analysis which builds on very similar assumptions. In particular, we shall take for granted, as [12] do for instance, that agents' argument moves should immediately improve their satisfaction with respect to the current situation of the debate. This work however departs from these previous proposals, in the sense that we address a case of *multiparty argumentation*. In this context, a number ($n > 2$) of agents exchange arguments on a common gameboard. No central computation of the whole system takes place, and no coordination between agents is assumed (even if they share the same view). The motivating applications we have in mind are for example online platforms allowing users to asynchronously modify the content of a collective debate. We want to study what outcomes will be reached with these type of interactions. This situation has received so far little attention and there are good reasons for that (see [4] for a discussion on the challenges raised by multiparty dialogues, and [16] for a recent study of multiparty persuasion in a specific framework). Firstly, it is not obvious to identify what would be the “correct” collective outcome in this case. In this paper we rely on a specific (natural in our case) *merged solution* [3] to assess the quality of the outcome. Secondly, the design of these protocols is made very difficult by the number of parameters to consider (think of several agents focused on possibly different issues), and renders the analysis of their formal properties challenging.

To keep things as simple as possible in this study, the following assumptions are made: (i) all the agents are focused on the same single issue (argument) of the debate (that is, agents evaluate how good is a state of the debate on the sole basis of the status of this specific argument); (ii) all the agents make use of the same argumentation semantics to evaluate both their private argumentation system and the situation on the common gameboard (specifically, we rely on Dung's grounded semantics [5]); and (iii) all the agents share the same set of arguments, but they may have different views on the attack relations between these arguments (this may result, *e.g.*, from agents being equipped with value-based argumentation systems [1] and ranking differently the values). While these restrictions are arguably severe, we will see that the resulting framework is already sufficiently rich to illustrate the variety of results that may be derived in the

Cite as: On the Outcomes of Multiparty Persuasion, E. Bonzon and N. Maudet, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 47-54.
Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

study of multiparty argumentation protocols.

The remainder of this paper is as follows. In the next section we provide the necessary background on argumentation semantics. Section 3 sets up the basic elements of our framework. The properties of the proposed protocol are studied in Section 4. Finally, Section 5 discusses related works and concludes, discussing possible extensions of the preliminary study proposed here.

2. BACKGROUND

2.1 Argumentation Systems

In this section, we briefly recall some key elements of abstract argumentation frameworks as proposed by Dung [5]. The exact content of arguments is left unspecified here. Instead, a (finite) set of arguments is given, as well as the different conflicts among them.

DEFINITION 1. An **argumentation system (AS)** is a pair $\langle A, R \rangle$ of a set A of arguments and a binary relation R on A called the **attack relation**. $\forall a, b \in A$, aRb (or $(a, b) \in R$) means that a **attacks** b (or b is **attacked** by a). An AS may be represented by a directed graph, called the **argumentation graph**, whose nodes are arguments and edges represent the attack relation.

From this argumentation graph, we can introduce some notions related to graph theory in order to characterize some properties of the argumentation system.

DEFINITION 2. Let AS be an argumentation system, and G be the argumentation graph associated. A **path** in G is a sequence of nodes such that from each node there is an edge to the next node in the sequence. A finite path has a first and a last node. An edge (b, c) is an **attack edge** (resp. **defense edge**) for an argument a iff there is an even-length (resp. odd-length) path from c to a .

Note that an edge can be both an attack and a defense edge. In Dung’s framework, the *acceptability* of an argument depends on its membership to some sets, called extensions. These extensions characterize collective acceptability.

DEFINITION 3. Let $AS = \langle A, R \rangle$ be an argumentation system. Let $S \subseteq A$. S is **conflict-free** for AS iff there exists no a, b in S such that aRb . S **collectively defends** an argument a iff $\forall b \in A$ such that bRa , $\exists c \in S$ such that cRb .

A set of arguments is *admissible* when it is conflict-free and each argument of the set is collectively defended by the set itself. Several *semantics for acceptability* have been defined in [5]. In what follows, we concentrate on the notion of *grounded semantics* which can be defined as follows:

DEFINITION 4. Let $AS = \langle A, R \rangle$ be an argumentation system. Let $S \subseteq A$. S is a **grounded extension** of AS iff S is the least fixed point of the characteristic function of AS ($F: 2^A \rightarrow 2^A$ with $F(S) = \{a \text{ such that } S \text{ collectively defends } a\}$).

Intuitively, a *grounded extension* contains all arguments which are not attacked, as well as the arguments which are defended (directly or not) by non-attacked arguments. There always exists a unique grounded extension. We shall denote by $\mathcal{E}(AS)$ the grounded extension of the system AS.

2.2 Merged Argumentation System

We now consider a set N of n agents. Each agent holds an argumentation system $AS_i = \langle A, R(i) \rangle$, sharing the same arguments A , but with possible conflicting views on attack relations between arguments (coming for instance from different underlying preferences). What should be the collective view in that case? To tackle this problem, we rely on the notion of a *merged* argumentation system [3]. In the specific case we discuss here, it turns out that a meaningful way to merge is to take the *majority argumentation system* where attacks supported by a majority of agents are kept (this corresponds to minimizing the sum of the edit distances between the AS_i and the merged system, see Prop. 41 in [3]). Assuming, on top of that, that ties are broken in favour of the absence of an attack allows to ensure the existence of a single such merged argumentation system, that we denote MAS_N .

DEFINITION 5. Let N be a set of agents and $\langle AS_1 \dots AS_n \rangle$ be the collection of their argumentation systems. The **majority argumentation system** is $MAS_N = \langle A, M \rangle$ where $M \subseteq A \times A$ and xMy when $|\{i \in N \mid (x, y) \in R_i\}| > |\{i \in N \mid (x, y) \notin R_i\}|$.

The corresponding *merged outcome* is denoted by $\mathcal{E}(MAS_N)$.

3. A PROTOCOL FOR FOCUSED AGENTS

We now turn to the following question: supposing that the agents of the system would not report to a central authority their whole argumentation system but instead contribute step-by-step in the debate, guided by their individual assessment of the current state of the discussion, and without coordination with other agents, what would be the outcome they would reach? For instance, can we guarantee that the merged outcome would always be reachable? To be able to formally answer this problem, we need of course to design a specific protocol and to make some assumption regarding agents’ preferences regarding the outcome.

3.1 Agents’ Preferences

We assume that agents are *focused* [14], that is, they concentrate their attention on a specific (same for all) argument. This argument is referred to as the *issue* d of the debate [11]. Unsurprisingly, agents want to see the acceptability status (under the grounded semantics) of the issue coincide in the debate and in their individual system. Thus we can see the debate as opposing two groups of agents: $CON = \{a_i \in N \mid d \notin \mathcal{E}(AS_i)\}$ and $PRO = \{a_i \in N \mid d \in \mathcal{E}(AS_i)\}$. If $X = PRO$ (resp. CON), we have $\bar{X} = CON$ (resp. PRO).

3.2 The Gameboard

Agents will exchange arguments via a common gameboard. The issue will be assumed to be present on this gameboard when the debate begins. The “common” argument system is therefore a *weighted argumentation system* [6] where the weight is simply a number equal in the difference between the number of agents who asserted a given attack and the number of agents who opposed it. We denote by $xR_\alpha y$ the fact that the attack has a weight α . Let $A(GB)$ be the set of all the arguments present on the gameboard. The collective outcome is obtained by applying the semantics used on the argumentation system $\langle A(GB), M \rangle$ where $M \subseteq A(GB) \times A(GB)$ and $xMy = \{xR_\alpha y \mid \alpha > 0\}$. In words, we

only retain those attacks supported by a (strict) majority of agents having expressed their view on this relation. Observe that following our tie-breaking policy we require the number of agents supporting the relation to strictly outweigh the number of agents who oppose it (*i.e.* in case of tie, the relation does not hold).

3.3 A Relevance-based Protocol

We now introduce our simple protocol which allows agents to exchange their arguments in order to agree on the status of a specific argument d , the issue of the dialogue. Let $AS^t(GB)$, $A^t(GB)$ and $R^t(GB)$ be respectively the argumentation system, the set of arguments and the set of attack relations on the gameboard after round t . The protocol indeed proceeds in rounds which alternate between the two groups of agents (PRO and CON). Within these groups though, no coordination takes place: the agents may for instance play asynchronously and the authority simply picks the first permitted and relevant move before returning the token to the other side. *Permitted* moves are simply positive assertions of attacks xRy (with $y \in A^t(GB)$), or contradiction of (already introduced) attacks (with $(x, y) \in R^t(GB)$). Note that arguments are progressively added on the gameboard via these attacks, and that it may not contain the whole set of arguments when the debate concludes. A move is *relevant* [10] at round t for a PRO agent (resp. CON agent) if it puts the issue back in (resp. drops the issue from) $\mathcal{E}(AS^t(GB))$. Furthermore, the protocol prevents the repetition of similar moves from the same agent. To account for this, each agent a_i is equipped with a set $RP_i^t \subseteq \{(x, y) | x, y \in A\}$ which contains the attack relations or the *non-attack* relation he has added on the gameboard at time t , in order to prevent him from adding twice the same relation. The proposed protocol is as follows:

- (1) Agents report their individual view on the issue to the central authority, which then assign (privately) each agent to PRO or CON.
- (2) The first round starts with the issue on the gameboard and the turn given to CON.
- (3) Until a group of agents cannot move, we have:
 - (a) agents independently propose moves to the central authority;
 - (b) the central authority picks the first (or at random) relevant move from the group of agents whose turn is active, update the gameboard, and passes the turn to the other group

When a (relevant) move is played on the gameboard, the following update operation takes place:

- (1) after an assertion xRy
 - if $xR_\alpha y \in R^t(GB)$ then $\alpha := \alpha + 1$
 - if $xR_\alpha y \notin R^t(GB)$ and $x, y \in A^t(GB)$, then the edge is created with $\alpha := 1$
 - otherwise (x is not present), then the node of the new argument is created and the edge is created with $\alpha := 1$
- (2) after a contradiction of xRy , we have $\alpha := \alpha - 1$

Note the asymetry here: introducing a new argument can only be done via a positive assertion, since it can never be relevant to contradict an attack referring to an argument that was not introduced already. The reader may remark that the value of α is binary if agents obey this protocol; however we discuss in Section 4.3 an extension where this is not necessarily the case.

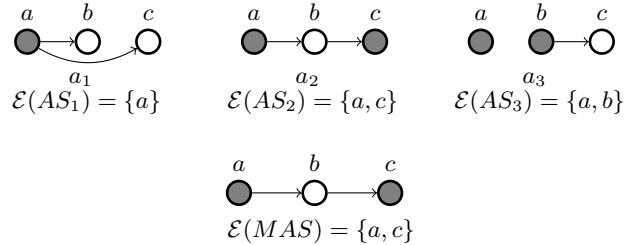
When (after a sequence σ of moves) a group of agents cannot move, we say that the gameboard is stable and we refer to $\mathcal{E}(AS(GB_{t \rightarrow \infty}^\sigma))$ (or simply $\mathcal{E}(AS(GB))$ when clear from the context) as the *outcome of the debate*.

3.4 Properties

The outcome $\mathcal{E}(AS(GB_{t \rightarrow \infty}^\sigma))$ resulting from a specific sequence of moves σ obeying this protocol will typically be compared with the result which would be obtained by merging the argumentation systems ($\mathcal{E}(MAS)$). We may want to ensure different properties, but we typically have:

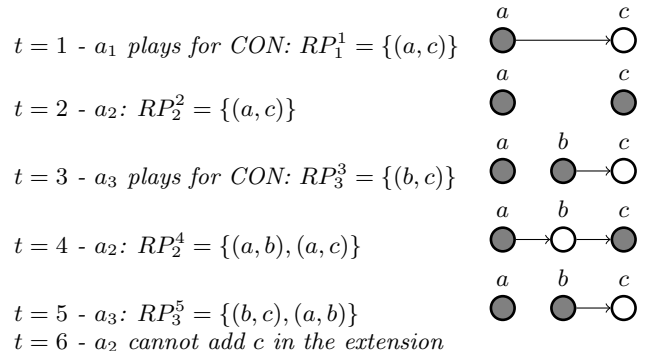
- *Termination*— trivially guaranteed by assuming finite argument systems and preventing move repetition.
- *Guaranteed convergence to the merged outcome*— requires *all* possible sequences of moves (in particular, regardless of the specific choice of the agent and of the move to pick, when several relevant moves are proposed to the authority) to converge to the merged outcome, that is $\forall \sigma d \in \mathcal{E}(AS(GB_{t \rightarrow \infty}^\sigma)) \leftrightarrow d \in \mathcal{E}(MAS_N)$
- *Reachability of the merged outcome*— requires *at least* one possible sequence of moves to reach the merged outcome, that is $\exists \sigma d \in \mathcal{E}(AS(GB_{t \rightarrow \infty}^\sigma)) \leftrightarrow d \in \mathcal{E}(MAS_N)$

EXAMPLE 1. *Let three agents with their argumentation systems, and the following merged argumentation framework:*



The issue of the dialogue is the argument c . We have $CON = \{a_1, a_3\}$, $PRO = \{a_2\}$. At the beginning, we have $RP_1^0 = RP_2^0 = RP_3^0 = \{\}$, $AS^0(GB) = \langle \{c\}, \{\} \rangle$ and $\mathcal{E}(AS^0(GB)) = \{c\}$.

A sequence of moves allowed by the protocol is the following:

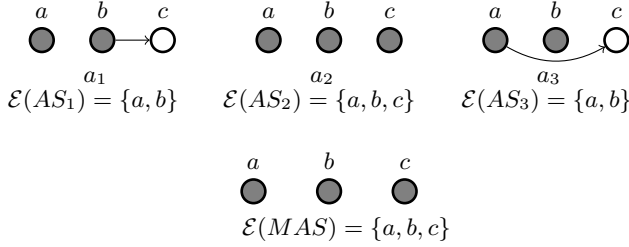


The game board is stable, we obtain $\mathcal{E}(AS(GB)) = \{a, b\}$.

The first interesting thing to observe on this simple example is the fact that the status of an issue in the merged argumentation system can contradict the opinion of the majority. This is discussed in [3]: if agents vote on extensions, the attack relations from which extensions are characterized are not taken into consideration, and a lot of significant information is not exploited.

Another important thing to note in this example is that PRO agents cannot ensure c in $\mathcal{E}(AS(GB))$. It is then impossible to guarantee convergence to the status of the issue obtained in the merged argumentation system. This is due to the fact that agent a_1 has no interest to play the attack relation (a, b) , which appears in the MAS. As studied in a different context by [12], this can be seen as a strategic manipulation by withholding an argument or an attack between arguments. But is it even possible to reach the merged outcome in this case? We leave it to the reader to check that this is not the case here. One may then think that the group with the highest number of agents will always win with our protocol. It is not the case, as shown by the fairly simple following example.

EXAMPLE 2. Let three agents with their argumentation systems, and the following merged argumentation framework:



The issue of the dialogue is the argument c . We have $CON = \{a_1, a_3\}$, $PRO = \{a_2\}$. Agents in CON can attack c in two ways: either a_1 can play bRc ; or a_3 can play aRc . But a_2 will be able to remove either attack, and CON agents will not have the possibility to counter-attack. We will obtain $\mathcal{E}(AS(GB)) = \{a, b, c\}$.

The two previous examples show that the characterization of the result obtained by debates following this protocol is not as simple as one can believe at first glance. We now introduce some useful and more sophisticated notions.

3.5 Global arguments-control graph

In order to characterize the status of the issue obtained by our protocol we will need the notion of *global arguments-control graph* (ACG). The idea here is to gather the attacks of all agents in the same argumentation graph, and then determine which group, PRO or CON, have the control over some path of this graph, and thus a possible way to reach its preferred outcome. To do so, we first need to define the notion of *control* over an attack relation:

DEFINITION 6. Let N be a set of agents, $\langle AS_1 \dots AS_n \rangle$ be the collection of their argumentation systems, and $L = \cup_{i \in 1 \dots n} R(i)$ be the union of all attack relations. Let $X \in \{CON, PRO\}$. Finally, let $add_{(a,b)} = \{a_i \in N \mid (a,b) \subseteq R(i)\}$, and $rem_{(a,b)} = \{a_i \in N \mid (a,b) \not\subseteq R(i)\}$.

- X has the **constructive control** of $(a,b) \in L$, denoted by $X^+(a,b)$, iff $|add_{(a,b)} \cap X| > |rem_{(a,b)} \cap \bar{X}|$, that is if the number of agents in X who can add (a,b) is greater than the number of agents in \bar{X} who can remove it.
- X has the **destructive control** of $(a,b) \in L$, denoted by $X^-(a,b)$, iff $|rem_{(a,b)} \cap X| \geq |add_{(a,b)} \cap \bar{X}|$, that is if the number of agents in X who can remove (a,b) is greater or equal than the number of agents in \bar{X} who can add it.

The following remarks are simple but useful: (1) It is impossible to have both $X^+(a,b)$ and $\bar{X}^-(a,b)$; (2) It is possible to have both $X^+(a,b)$ and $X^-(a,b)$; (3) A minority group cannot have constructive and destructive control of an edge: if $|X| < |\bar{X}|$, it is impossible to have both $X^+(a,b)$ and $X^-(a,b)$; (4) If there is not $X^-(a,b)$ (resp. $X^+(a,b)$), then there is $\bar{X}^+(a,b)$ (resp. $\bar{X}^-(a,b)$).

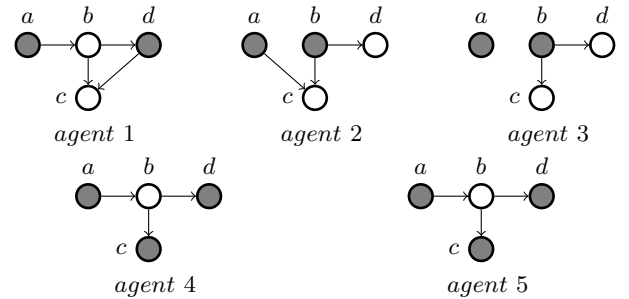
Observe that the notion of destructive control intuitively says that a group has the control to overweight any possible attempt to establish a given relation. This of course vacuously holds when no agent from the other group supports the relation at all, in which case the relation is not even playable.

DEFINITION 7. Let N be a set of agents and $\langle AS_1 \dots AS_n \rangle$ be the collection of their argumentation systems. We will say that $(a,b) \in \cup_{i \in 1 \dots n} R(i)$ is **playable** by a $X \in \{PRO, CON\}$, denoted by $X_\bullet(a,b)$, iff there is an $a_i \in X$ such that $(a,b) \in R(i)$.

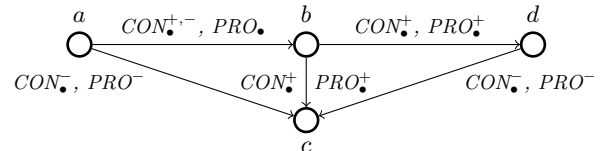
For the sake of readability, we will only specify the information about playability when it is relevant.

DEFINITION 8. Let N be a set of agents and $\langle AS_1 \dots AS_n \rangle$ be the collection of their argumentation systems. The **global arguments-control graph** is $ACG_N = \langle A, L \rangle$ is constructed as follow: (1) $L = \cup_{i \in 1 \dots n} R(i)$ (2) Label each $(a,b) \in L$ by the information about control and playability for each group $X \in \{PRO, CON\}$.

EXAMPLE 3. Five agents have the following argumentation systems:



The issue of the dialogue is the argument c . We have $CON = \{a_1, a_2, a_3\}$, $PRO = \{a_4, a_5\}$. The global arguments-control graph is the following:



4. PROPERTIES

We now discuss three distinct properties that we wish to analyze on the basis of the ACG: (i) who wins the debate?, (ii) does the outcome of the debate coincide with that of the merged system?, and (iii) is it useful to allow moves that reinforce previous moves?

4.1 Who wins the debate?

The first question that we address is whether an omniscient observer would know *a priori* which group of the debate could possibly or necessarily win the debate, in particular whether some debates are “open” (*i.e.* not pre-determined [8]).

DEFINITION 9. *We will say that the issue of the debate is a **possible outcome** for a group X if this group has a possibility to set the acceptability status of this argument to coincide in the debate and in their individual system. The issue is a **necessary outcome** for X iff this issue is not a possible outcome for \bar{X} .*

DEFINITION 10. *A **path for d controlled by CON** is an odd-length path from x to d such that (i) CON has constructive control on all the attack edges for d , and (ii) CON has destructive control on all the defense edges for d attacking x .*

Note that condition (ii) covers in particular the case where the first node x is not attacked. Controlling a path is not enough since alternative defenses may exist. By extension, we then define the notion of a tree controlled by CON .

DEFINITION 11. *A **tree for d controlled by CON** is a tree such that (i) d is the root, (ii) all the paths from the leaves to d are controlled by CON , and (iii) for any attack edge yRx of the tree, it contains all the defense edges zRy such that $PRO^+(z, y)$.*

This gives us a condition guaranteeing that a favourable outcome can be attained by CON .

PROPOSITION 4.1. *If there exists a tree for d controlled by CON , then the issue d is a possible outcome for CON .*

PROOF. (Sketch.) Observe that because we have a tree no edge can be played both as an attack and a defense edge. Then CON can certainly win by making sure that all the attack moves of the tree are placed, since it can respond to any possible defense edge of the tree on which PRO has constructive control, and it can certainly remove any other defense edge which could be played by PRO (because it must hold the destructive control on these edges). ■

A couple of remarks are in order here. If the ACG itself happens to be a tree, then the above condition is necessary and sufficient to guarantee that the outcome is necessary for CON . However, in general, things turn out to be much more involved. First this condition is not necessary for the outcome to be possible for CON : this group of agents may win in absence of such a tree (in fact, even in absence of a single path controlled by himself). This may look counter-intuitive, but the reason lies on the fact that the control of an edge may be gained during the debate in specific circumstances. We do not elaborate on this point here but a related

observation is developed in Section 4.3. Secondly, this condition is not strong enough to guarantee that the issue is a necessary outcome for CON . Indeed, in absence of coordination, agents of CON may not play the moves of the tree only. And there are cases where this may make d a possible outcome for PRO . To see why this may be the case, recall that an edge may be both an attack and a defense edge for the same issue d , as it may appear on several distinct paths. When that happens, this edge may be used as an attack edge, preventing the deployment of the path controlled by CON . The following notions of *switch* captures this.

DEFINITION 12. *An edge (x, y) on a path P is a **switch** for d if (i) it is a defense for d on P , (ii) it is playable by CON , (iii) there exists an even-length path from y to d such that all the attack edges are playable by CON and all the defense edges are playable by PRO . So it is also a potential attack for d via a different path.*

Essentially, what this definition says is that there is a possibility that this edge (x, y) may be played as an attack by CON . As mentioned before this may harm CON own line of attack. Following this, we say that there exists a *switch for path P for d controlled by CON* if there exists a defense edge for d attacking x (the first node of P) that is a switch. Each path in which a switch for P is an attack edge is called a *switch path of P* .

We are now in a position to informally state some conditions under which d may not be a necessary outcome for CON despite the existence of a tree controlled by himself. In fact it is the case when there exists a set of switches \mathcal{S} such that: (i) for any tree t for d controlled by CON , there exists a switch belonging to \mathcal{S} for a path (of t) for d controlled by CON ; (ii) there must exist a sequence of moves such that (1) all the switches in \mathcal{S} are actually played, and (2) PRO has the destructive control over an attack edge of each resulting switch path; (iii) there must exist a sequence of moves such that all the switches in \mathcal{S} are maintained.

It may not be immediately clear to the reader why the mere existence of switches —Cond (i)— does not imply the fact that they can be played —Cond (ii.1): after all, the definition requires a path of playable moves reaching the switch to exist. The subtlety lies on the fact that these paths may interact when they share some arguments. In this case, the existence of a path may preclude other paths to be played. Intuitively, Cond (iii) caters for the fact that a switch may be “patched” by CON if he manages to append an odd-length path right behind the switch.

The next question is whether these conditions can be simply expressed on the basis of the ACG. For (i) and (ii.2) this is obvious. For (ii.1) and (iii) this is more challenging because the definition refers to possible sequences of moves. We will rely instead on sufficient conditions:

PROPOSITION 4.2. *The issue d may not be a necessary outcome for CON if there exists a set of switches \mathcal{S} such that: (i) for any tree t for d controlled by CON there exists a switch belonging to \mathcal{S} for a path (of t) for d controlled by CON ; (ii) there exists a set of switch paths \mathcal{P} for \mathcal{S} such that these paths do not share any arguments (except d), and PRO has destructive control over an attack edge of each of these paths; (iii) for all switches $(x, y) \in \mathcal{S}$, there does not exist any even-length path P reaching d meeting x , such that $[x, d]$ constitutes an odd-path length, and such that CON has*

the constructive control on all attack edges, and PRO has the constructive control on all the defense edges.

PROOF. (Sketch.) Cond (i) ensures that at least an attack path of each tree controlled by CON can be potentially switched. Cond (ii) suffices to guarantee that the switch paths are independent, so all the switches can actually be played, and the switch paths subsequently cut. As for (iii), observe that there are two ways to render a switch (xRy) ineffective. Either by simply removing it, but this necessarily requires a path (leading to the issue) to meet the node y of the switch (so that playing xRy would be relevant). This is impossible. Or by appending an odd-length path (a “patch”) on x of the switch, such that a move meeting the node y of the switch could now be played. This can only happen when there exists an even-length path reaching d meeting x , such that $[x, d]$ constitutes an odd-path length, and such that CON has the constructive control on all attack edges, and PRO has the constructive control on all the defense edges; otherwise there is necessarily a possibility that this path does not reach x . ■

Note that for (ii) it may be the case that switch paths, even interacting, allow the switch to be played. For (iii) it may be the case, on the other hand, that even if such paths leading to patches do exist, they could not be played because they are interacting in a way that makes them mutually exclusive. What this discussion suggests is that obtaining a full characterization of outcomes is certainly very challenging in the general case. It provides however a simple way to construct examples of debates that are indeed open.

EXAMPLE 3 CONT. We easily see that the issue c is a **possible outcome** for the agents in CON: CON can attack c with b . Then, the only possible move for PRO is to defend d with aRb . However, CON can remove this attack, and PRO has no other move.

But c is also a **possible outcome** for PRO: CON can start with dRc , which is playable by a_1 . Then, a_5 will defend with bRd , and a_1 counter-attack with aRb . If the next move of PRO is to remove dRc , then CON has no other move left: it cannot add the attack bRc , as it is defended by a ; and it cannot remove the edge (a, b) as it does not drop c from the extension. In this case, (a, b) is a switch and the merged outcome is then (only) reachable.

4.2 Does it coincide with the MAS?

The next step here is to characterize the convergence and/or the reachability of the merged outcome. We have already seen that the merged outcome is not always reachable, but is it possible to find some case for which it is? To answer this question, we first need the following lemma.

LEMMA 1. Let N be a set of agents, $ACG_N = \langle A, L \rangle$ be the global arguments-control graph and $MAS_N = \langle A, M \rangle$ be the merged argumentation system. If there is no edge $(a, b) \in L$ such that a group $X \in \{CON, PRO\}$ has the constructive and destructive control of (a, b) , then all the edges controlled constructively in the ACG belong to the MAS, whereas all the edges controlled destructively in the ACG do not belong to the MAS.

PROOF. By remark (4), we know that either $X^+(a, b)$ and $\bar{X}^+(a, b)$, or $X^-(a, b)$ and $\bar{X}^-(a, b)$. Take the case of constructive control: we have $|add_{(a,b)} \cap X| > |rem_{(a,b)} \cap \bar{X}|$ and

$|rem_{(a,b)} \cap X| < |add_{(a,b)} \cap \bar{X}|$. As $X \cap \bar{X} = \{\}$, we have $|add_{(a,b)}| > |rem_{(a,b)}|$. Then, by definition of the merged argumentation system, we know that $(a, b) \in M$ (that is (a, b) is an edge of the MAS). The case of destructive control is similar. ■

This lemma leads to the following proposition.

PROPOSITION 4.3. Let N be a set of agents, and $ACG_N = \langle A, L \rangle$. If there is no edge $(a, b) \in L$ such that $X^{+,-}(a, b)$, then the merged outcome is reachable.

PROOF. We know from Lemma 1 that all the edges controlled constructively in the ACG belong to the MAS, whereas all the edges controlled destructively in the ACG do not belong to the MAS. Let d be the issue of the debate.

(1) Let us assume that $d \in \mathcal{E}(MAS)$. Thus, for all $x \in A$ such that xMd , there is an even-length path $P = (x_1, x_2, \dots, x, d)$ which defends d . As all these edges belong to the MAS, we know from Lemma 1 that they belong to the ACG, and that they are controlled constructively by PRO and by CON. Thus, CON can play all the attack edges of P , whereas PRO can defend d by adding all the defense edges of P . As x_1 is not attacked in the MAS, there are two possibilities in the ACG:

- Either x_1 is not attacked in the ACG. In this case, CON can not attack x_1 , and then has no possibility to drop d from $\mathcal{E}(AS(GB))$.
- Or there is an attack edge (y, x_1) in the ACG. As this edge is not in the MAS, we know that $PRO^-(y, x_1)$. So, PRO can remove this edge and then ensure that $d \in \mathcal{E}(AS(GB))$

As this reasoning holds for all defense path in the MAS, and is playable with our protocol, d is reachable.

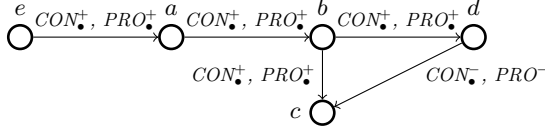
(2) Let us assume now that $d \notin \mathcal{E}(MAS)$. So, there is an odd-length path $P = (x_1, x_2, \dots, x, d)$ in the MAS which attacks d . As all these edges belong to the MAS, we know from Lemma 1 that they belong to the ACG, and that they are controlled constructively by PRO and by CON. Thus, CON can play all the attack edges of P , whereas PRO can defend d by adding all the defense edges of P . As x_1 is not attacked in the MAS, there are two possibilities in the ACG:

- Either x_1 is not attacked in the ACG. In this case, PRO can not attack x_1 , and then has no possibility to put d in $\mathcal{E}(AS(GB))$.
- Or there is an attack edge (y, x_1) in the ACG. As this edge is not in the MAS, we know that $CON^-(y, x_1)$. So, CON can remove this edge and then ensure that $d \notin \mathcal{E}(AS(GB))$

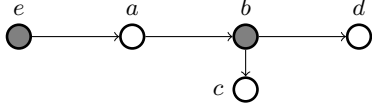
As this path is playable with our protocol, we know that d is reachable. ■

Note that we can only ensure the reachability. The following example shows that we do not have guaranteed convergence.

EXAMPLE 4. Consider the following global arguments-control graph and merged argumentation systems, where c is the issue.



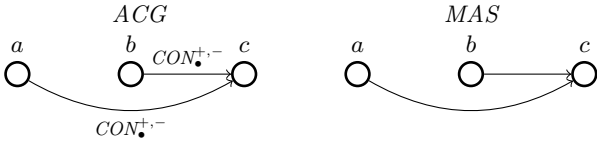
From Lemma 1, we know that the graph of the merged argumentation system is the following:



Thus, $c \notin \mathcal{E}(MAS)$. However, if we suppose that the edge (d, c) is playable for CON, c is a possible outcome for PRO: CON can start by adding dRc . Then, PRO will defend with bRd , and CON counter-attack with aRb . If the next move of PRO is to remove the attack dRc , then CON has no other move left: it cannot add the attack bRc , as it is defended by a ; and it cannot remove the edge aRb as it does not drop c from the extension. But c is also a possible outcome for CON: the merged outcome is (only) reachable.

Another important remark is that the converse of Proposition 4.3 is false: as shown by the following example, it is possible for a group to have constructive and destructive control of an edge of the global arguments-control graph, and to ensure the reachability of the merged outcome.

EXAMPLE 5. Consider the following global arguments-control graph, where c is the issue.



In this graph, CON has the constructive and destructive control over two edges, and the merged outcome is reachable: the outcome is necessary for CON, and $c \notin \mathcal{E}(MAS)$.

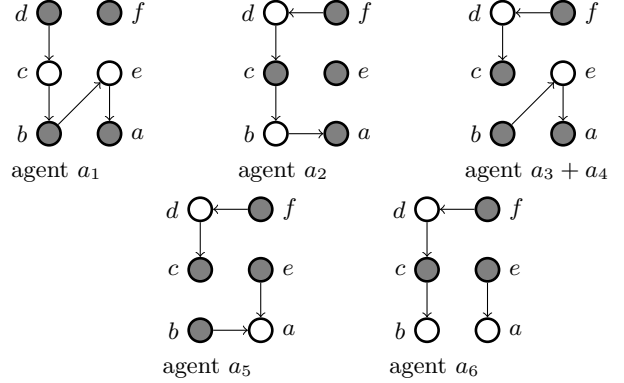
4.3 Is it useful to allow reinforcement?

A natural extension is to consider that a move may also be relevant as long as it *reinforce* (or symmetrically *weakens*) an edge which, if deleted and all other things being equal, would change the status of the issue. Essentially, besides the relevant moves as defined in the previous section, this would allow agents to augment the weight of an existing attack, and we refer to this as a *reinforcement move*. Symetrically, agents may weaken an attack even if it does not directly delete it, and we refer to this as a *weakening move*. This extended protocol would allow any number of such relevant moves during a group's turn, but (as before) would only switch to the other side after a change of the current status of the issue. However, the following proposition tells us that it is not beneficial for an agent to play reinforcement moves. Worse, and rather counter-intuitively, it can actually be damaging for agents to do so.

PROPOSITION 4.4. Let D_1 be sequences where no agent plays reinforcement or weakening moves, and D_2 be sequences such that X only may play reinforcement moves (but \bar{X} may play weakening moves).

If $X = \text{PRO}$ (resp. CON), then (i) for any $\sigma_2 \in D_2$ with $d \in \mathcal{E}(AS(GB_{t \rightarrow \infty}^{\sigma_2}))$ (resp. $d \notin \mathcal{E}(AS(GB_{t \rightarrow \infty}^{\sigma_2}))$), there exists $\sigma_1 \in D_1$ such that $AS(GB_{t \rightarrow \infty}^{\sigma_1}) = AS(GB_{t \rightarrow \infty}^{\sigma_2})$. Further (ii) there exists $\sigma_2 \in D_2$ with $d \notin \mathcal{E}(AS(GB_{t \rightarrow \infty}^{\sigma_2}))$ (resp. $d \in \mathcal{E}(AS(GB_{t \rightarrow \infty}^{\sigma_2}))$), such that for any $\sigma_1 \in D_1$ it is not the case that $AS(GB_{t \rightarrow \infty}^{\sigma_1}) = AS(GB_{t \rightarrow \infty}^{\sigma_2})$.

PROOF. (Sketch.) We show (ii) by constructing an example where an agent *loses* some destructive control by using reinforcement. It involves 6 agents.



The issue of the debate is a . There are four agents PRO and two agents CON. The key to the analysis is to see that PRO agents initially hold constructive and destructive control on (c, b) . Now compare the following sequences of moves. In the first one, a_5 plays bRa . Then a_1 plays cRb and a_2 reinforces this move. At this point, PRO loses its destructive control on (c, b) . Assume a CON agent plays dRc . a_1 can remove bRa . Then a_5 can play eRa , a_3 can defend with bRe . Now, with a CON agent playing fRd , the debate is doomed with the issue *out*. In the alternative case where no reinforcement is played, we are in the case discussed in the first protocol: PRO can remove the attack cRb , and win the debate. ■

This result tells us that in the absence of coordination, agents are better off employing moves that are directly relevant, hence adopting a “wait and see” approach. Still, using reinforcement moves may prove useful in practice, in contexts where the debate is limited: for instance, agents may be impressed by seemingly large majorities and avoid these issues to concentrate on some other ones.

5. RELATED WORKS AND CONCLUSION

As already mentioned, our work is close in spirit to the work of Rahwan and Larsson [12]. An important difference with our approach though is that agents control the arguments they can advance in the debate, but that no disagreement takes place regarding the attack relations between these arguments. Another recent proposal of great interest is that of Caminada and Pigozzi [2]. The authors propose different procedures to aggregate different labellings for a given

argumentation system into a collective one. The property they want to ensure is that the obtained collective outcome is in some sense compatible with the individual ones. A related contribution of Rahwan and Tohmé [14], which investigates the same question and derives general conclusions on the possibility (or impossibility) to perform such an aggregation, under classical assumptions. As mentioned already, these approaches assume that agents agree on the underlying argumentation system, even though they may have different views on the preferred labelling. Finally, a multiparty protocol for agents equipped with defeasible logic reasoning abilities is investigated in [9]. Each agent initially puts forward an initial claim and the protocol lets iteratively each agent defend his claim or attack the claim of opposing agents (by relying on a sophisticated technique to identify the most effective counter-arguments).

In our proposal, a multiagent protocol regulates the exchange of arguments among focused agents on the basis of the relevance of the moves as proposed by [10]. Although all the agents share the same set of arguments, they may have different views on the attack relations among these arguments. In case of discrepancy on a relation we have opted for a majoritarian approach: the side supported by the highest number of agents wins (more sophisticated approaches are discussed in [15]). Furthermore, even though agents exchange arguments on a common gameboard, it is important to note that no central authority gets to know the whole argumentation system of each agent. We have investigated some formal properties of this protocol. In particular, we have shown that there are cases where the outcome is not entirely pre-determined from the initial situation, and discussed non-trivial circumstances which may give rise to such debates (based on different notions of control of attacks by a group of agents). We have also given conditions under which the merged outcome can be reached, and discussed a natural extension of the protocol where moves can be reinforced (but showed that agents can only be worse off by using these extended set of moves).

A natural follow-up of this work would be to provide some insights regarding how often the debate is indeed open or how often coincidence with merged outcome is observed. Experiments could prove instructive in this respect. As for possible extensions of this work, it is clear that any relaxations of these assumptions brings about some complexity. If agents do not focus on a single issue, among other things, we may not simply distinguish two groups PRO and CON, and it becomes necessary to specify complex preferences over combinations of issues. If we relax the assumption of the set of arguments being shared, we then need to deal (see [3]) with the complex problem of how agents would react in the presence of arguments they were not aware of before. In the perspective of modeling practical debate platforms as mentioned in the introduction, all these aspects will require of course careful study.

Acknowledgements.

We thank Dionysios Kontarinis, Sébastien Konieczny, and the anonymous reviewers, for their numerous and detailed comments.

6. REFERENCES

- [1] T. Bench-Capon. Value-based argumentation frameworks. In *Proc. of the 9th Int. Workshop on Non-Monotonic Reasoning (NMR'02)*, pages 443–454, 2002.
- [2] M. Caminada and G. Pigozzi. On judgment aggregation in abstract argumentation. *Journal of Autonomous Agents and Multiagent Systems*, 22:64–102, 2011.
- [3] S. Coste-Marquis, C. Devred, S. Konieczny, M.-C. Lagasque-Schiex, and P. Marquis. On the Merging of Dung’s Argumentation Systems. *Artificial Intelligence*, 171:740–753, 2007.
- [4] F. P. Dignum and G. A. Vreeswijk. Towards a testbed for multi-party dialogues. In *Advances in Agent Communication*, volume 2922 of *Lecture Notes in Computer Science*, pages 1955–1955. Springer, 2004.
- [5] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-persons games. *Artificial Intelligence*, 77:321–357, 1995.
- [6] P. Dunne, A. Hunter, P. McBurney, S. Parsons, and M. Wooldridge. Inconsistency tolerance in weighted argument systems. In *Proc. of the 8th Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'09)*, pages 851–858, 2009.
- [7] R. Loui. Process and policy: Resource-bounded nondemonstrative reasoning. *Computational Intelligence*, 14(1):1–38, 2002.
- [8] S. Parsons, M. Wooldridge, and L. Amgoud. Properties and complexity of some formal inter-agent dialogues. *Journal of Logic and Computation*, 13(3):347–376, 2003.
- [9] D. H. Pham, G. Governatori, and S. Thakur. Extended defeasible reasoning for common goals in n-person argumentation games. *Journal of Universal Computer Science*, 15(13):2653–2675, 2009.
- [10] H. Prakken. Coherence and flexibility in dialogue games for argumentation. *Journal of Logic and Computation*, 15:347–376, 2005.
- [11] H. Prakken. Formal systems for persuasion dialogue. *Knowledge Engineering Review*, 15:1009–1040, 2005.
- [12] I. Rahwan and K. Larson. Pareto optimality in abstract argumentation. In *Proc. of the 23rd Conference on Artificial Intelligence (AAAI'08)*, pages 150–155, 2008.
- [13] I. Rahwan and K. Larson. *Argumentation and Game Theory*, chapter Argumentation in Artificial Intelligence, pages 321–339. Springer, 2009.
- [14] I. Rahwan and F. A. Tohmé. Collective argument evaluation as judgement aggregation. In *Proc. of the 10th Int. Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pages 417–424, 2010.
- [15] F. A. Tohmé, G. A. Bodanza, and G. R. Simari. Aggregation of attack relations: A social-choice theoretical analysis of defeasibility criteria. In *Proc. of the 5th Int. Symposium on the Foundations of Information and Knowledge Systems*, volume 4932 of *LNCS*, pages 8–23. Springer, 2008.
- [16] M. Wardeh, T. Bench-Capon, and F. Coenen. Multi-party argument from experience. In *Proc. of the 6th Int. Workshop on Argumentation in Multiagent Systems (ArgMas'09)*, pages 216–235, 2009.

Arbitrators in Overlapping Coalition Formation Games

Yair Zick

Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
yair0001@ntu.edu.sg

Edith Elkind

Division of Mathematical Sciences
School of Physical and Mathematical Sciences
Nanyang Technological University, Singapore
eelkind@ntu.edu.sg

ABSTRACT

Overlapping Coalition Formation (OCF) games [3, 4] are cooperative games where the players can simultaneously participate in several coalitions. Capturing the notion of stability in OCF games is a difficult task: a player may deviate by abandoning some, but not all of the coalitions he is involved in, and the crucial question is whether he then gets to keep his payoff from the unaffected coalitions. In [4] the authors introduce three stability concepts for OCF games—the *conservative*, *refined*, and *optimistic* core—that are based on different answers to this question. In this paper, we propose a unified framework for the study of stability in the OCF setting, which encompasses the concepts considered in [4] as well as a wide variety of alternative stability concepts. Our approach is based on the notion of an *arbitrator*, which can be thought of as an external party that determines payoff to deviators. We give a complete characterization of outcomes that are stable under arbitration. In particular, our results provide a criterion for the outcome to be in the refined or optimistic core, thus complementing the results in [4] for the conservative core, and answering questions left open in [4]. We also introduce a notion of the nucleolus for arbitrated OCF games, and argue that it is non-empty. Finally, we extend the definition of the Shapley value [12] to the OCF setting, and provide an axiomatic characterization for it.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent Systems

General Terms

Theory

Keywords

Overlapping Coalition Formation, Core, Nucleolus, Shapley Value

1. INTRODUCTION

Cooperation among agents plays a crucial role in the functioning of multi-agent systems. Therefore, developing a better understanding of coalition formation processes is an important research agenda in the multiagent community, and a lot of recent research effort has been spent on the design and analysis of cooperation mechanisms

Cite as: Arbitrators in Overlapping Coalition Formation Games, Yair Zick and Edith Elkind, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 55-62.
Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

for realistic multi-agent environments [15, 10, 5, 14, 8]. In such environments, agents are often selfish, and therefore need to be given incentives to act together and share the benefits of cooperation in a fair manner. Cooperative game theory [9, 18] provides the theoretical underpinnings for the study of such settings. Traditionally, it models a multiagent system as a (transferable-utility) game. Such a game can be described by its *characteristic function*, which for every set of agents specifies the profit that these agents can attain by working together. The agents are expected to split into teams, i.e., form a *coalition structure*; the profits of each team are then distributed among its members.

Remarkably, the traditional model assumes that each agent participates in exactly one coalition. However, this is often not the case in real-life settings, where agents form multiple coalitions on the fly in order to perform a specific task and only devote part of their attention and resources to each such coalition. Indeed, Shehory and Kraus in their seminal paper [14] already mention that agents can benefit from forming overlapping coalitions, and propose algorithms for iterative formation of an overlapping coalition structure for their setting. This line of work has been continued by Dang et al. [5], where the authors consider overlapping coalition formation in sensor networks. However, these papers assume that agents are fully cooperative, and will always form the socially optimal (overlapping) coalition structure. While this assumption is appropriate for the specific scenarios considered in these papers, in general, agents may want to maximize their own welfare, and a fully expressive model for overlapping coalition formation should take incentive issues into account.

Recently, Chalkiadakis et al. [3, 4] addressed this problem by proposing a game-theoretic model for overlapping coalition formation. In their model, each agent is endowed with a certain amount of resources, which he is free to distribute across multiple coalitions. The value of such (partial) coalition is determined both by the identities of agents that participate in it and the amount of resources that they contribute. Chalkiadakis et al. [3, 4] focus on the study of stability in their model. Compared to the non-overlapping setting, the stability of an overlapping coalition structure is a delicate issue: if an agent is participating in several projects at once and decides to withdraw all or some of her contributions from one of them, can she expect to continue to receive the payoff from the coalitions that were not harmed by the deviation? In [4], the authors propose three different stability concepts—the conservative core, the refined core, and the optimistic core—that correspond to three possible ways of answering this question. Briefly, under the conservative core the deviators do not expect to get any payoffs from their coalitions with non-deviators. In contrast, in the refined core they continue to get payoffs from coalitions not affected by the deviation. Finally, in the optimistic core the deviators may get

some payoffs from an affected coalition, as long as they continue to contribute to it, and the members of that coalition were able to regroup and focus on a different task so that each non-deviator still gets as much profit as before from that coalition.

While the three concepts of the core proposed in [4] all correspond to reasonable reactions to deviation, this list is by no means exhaustive. For instance, a player may want to punish the deviators and refuse to cooperate with them altogether as soon as they lower the value of one of the coalitions he is involved in, even if other coalitions between that player and the deviator remain unaffected. Alternatively, the players may form a social network, and stop collaborating with a deviator if his behavior harmed one of their friends. Yet another possibility is that a central authority imposes a fine on each of the deviators, making the deviation costly.

In this paper, we propose a stability concept that captures all of the scenarios considered above. Our approach is based on the notion of an *arbitrator*: a function that takes the description of a deviation as an argument and returns the payoff that the deviators receive from each coalition. Different arbitrators correspond to different sets of stable outcomes, or *arbitrated cores*. We show that the three core concepts proposed in [4] can be viewed as special cases of our model. Further, paper [4] characterizes the set of outcomes that belong to the conservative core. We extend this characterization to all arbitrated cores. In particular, this allows us to characterize the outcomes in the refined core and the optimistic core, thus answering the open question proposed in [4].

Now, while the core is an attractive stability concept, it is known that some games have an empty core. This is true even in the overlapping model for most realistic arbitrators. Thus, it is desirable to have a solution concept that identifies “the most stable” (overlapping) coalitions, yet is guaranteed to be non-empty. In the non-overlapping setting, this role is fulfilled by the nucleolus [11]. Motivated by this intuition, we introduce a concept of nucleolus for the OCF setting, and demonstrate that it is always non-empty. However, in contrast to the traditional model, we show that in OCF games the nucleolus may contain more than one outcome.

Finally, we extend the definition of the Shapley value [12] to the OCF setting. Just as in the classic case, we present a set of natural axioms, and demonstrate that our variant of the OCF Shapley value is characterized by these axioms.

The rest of this paper is organized as follows. After presenting the necessary background material in Section 2, we introduce the notion of arbitration and arbitrated core in Section 3, and present our characterization of the outcome in the arbitrated core in Section 4. Section 5 focuses on the nucleolus for OCF games, and Section 6 describes our extension of the Shapley value to the OCF setting. Section 7 presents our conclusions and suggests directions for future work.

2. PRELIMINARIES

We begin by describing our notation and the formal model of OCF games. Our definitions mostly follow those in [4]. We, however, describe deviation in a manner more conducive to our analysis.

Notation Throughout the paper, we write $N = \{1, \dots, n\}$. Given a vector $\mathbf{x} = (x^1, \dots, x^n) \in \mathbb{R}^n$ and a set $S \subseteq N$, we write $\mathbf{x}(S) = \sum_{i \in S} x^i$, $\mathbf{x}|_S$ equals \mathbf{x} on the S coordinates and is 0 otherwise, and e^S is the indicator vector of S .

Classic TU Cooperative Games A *transferable utility (TU) cooperative game* is defined by a set of players N and a *characteristic function* $u : 2^N \rightarrow \mathbb{R}$ with $u(\emptyset) = 0$. The set of *feasible payoffs* for a game $G = (N, u)$ is the set of all vectors $\mathbf{x} \in \mathbb{R}^n$ such that $\mathbf{x}(N) = u(N)$. A *solution concept* is a function that assigns

every TU game $G = (N, u)$ a set of feasible payoff vectors. A solution concept that assigns G a single point is called a *value*. For a detailed discussion of solution concepts and their axiomatization see [9], Section 2.3, pp. 19–25.

OCF Games Let $N = \{1 \dots n\}$ be a set of agents. A *partial coalition* of players in $S \subseteq N$ is a vector $\mathbf{c} \in [0, 1]^n$, where $c^i = 0$ for all $i \notin S$. That is, each player in S may contribute a fraction of their resources to \mathbf{c} . In what follows, we will omit the word “partial”, and refer to vectors in $[0, 1]^n$ as *coalitions*.

DEFINITION 2.1. An OCF game $G = (N, v)$ is given by a set of players N and a characteristic function $v : [0, 1]^n \rightarrow \mathbb{R}$ assigning a real value to each partial coalition; we require $v((0)^n) = 0$.

A *coalition structure* over N is a $n \times k$ matrix $CS = (\mathbf{c}_1, \dots, \mathbf{c}_k)$, where k is the number of coalitions. We require that for all $i \in N$ it holds that $\sum_{j=1}^k c_j^i \leq 1$. This means that CS is a valid division of players’ resources. Coalition structures over subsets of N are defined in a similar manner. Throughout the paper, we will assume that v is monotone; thus, we can assume that all players would want to invest all their resources, i.e. $\sum_{j=1}^k c_j^i = 1$; such coalition structures are called *efficient*. We denote the set of all possible coalition structures over S as CS_S . We also overload notation and define $v(CS) = \sum_{j=1}^k v(\mathbf{c}_j)$. Two coalition structures $CS = (\mathbf{c}_1, \dots, \mathbf{c}_k)$ and $CS' = (\mathbf{d}_1, \dots, \mathbf{d}_k)$ are *equivalent* if there is some permutation σ such that for all $1 \leq j \leq k$, $\mathbf{c}_j = \mathbf{d}_{\sigma(j)}$.

Similarly to [4], we would sometimes like to limit the maximum number of coalitions players can form; indeed, oftentimes an agent who gives less than a certain fraction of her resources to a coalition can no longer contribute to a coalition. If the number of coalitions is limited by $U \in \mathbb{N}$ we say that the game G is *U-finite*.

For any $CS \in CS_S$, we call the vector $w(CS) = \sum_{j=1}^k \mathbf{c}_j$ the *weight vector* of CS . Note that $w(CS) \in [0, 1]^n$, and if CS is efficient then $w(CS)$ is the indicator vector for the set S .

For each $S \subseteq N$ we denote by $v^*(S)$ the maximum value achievable by S : $v^*(S) = \sup\{v(CS) \mid CS \in CS_S\}$. Note that (N, v^*) can be viewed as a classic TU cooperative game; we will refer to (N, v^*) as the *crisp analogue* of G . We extend the function v^* to partial coalitions by setting $v^*(\mathbf{c}) = \sup\{v(CS) \mid w(CS) \leq \mathbf{c}\}$; v^* is the *superadditive cover* of v . Note that $v^*(e^J) = v^*(J)$. We remark that we borrow the term “crisp” from Aubin, who introduced the concept of fuzzy games (and their crisp analogues) in [1]. Like OCF games, fuzzy games are also defined by functions from $[0, 1]^n$ to \mathbb{R} . However, they are based on very different intuition, and, in particular, employ a very different notion of stability. We refer the reader to [4] for a detailed discussion of the differences between OCF games and fuzzy games.

It is often useful to think of the agents as using their resources to complete a given set of tasks. Such games are described in [4] and are called *Threshold Task Games (TTGs)*. A TTG comprises of a finite list of tasks, $T = \{t_1, \dots, t_k\}$, each t_i requires some weight $w(t_i) \geq 0$ for its completion, and gives a certain payoff $p(t_i) \geq 0$. Each player i has some weight $w_i \geq 0$ that he may allocate to the completion of any task. The worth of a coalition is

$$v(\mathbf{c}) = \max\{p(t_i) : w(t_i) \leq \sum_{i=1}^n c^i w_i\}.$$

We say that a function v has *Efficient Coalition Structure (ECS) property* if for any $J \subseteq N$ and any $\mathbf{w}_J \leq e^J$ there exists a coalition structure $CS_J \in CS_J$ such that $v^*(\mathbf{w}_J) = v(CS_J)$. All U-finite continuous functions have the ECS property, since the set of all coalition structures with weight less than \mathbf{w}_J is compact (due to U-finiteness); v is continuous and hence achieves a maximum over

a compact set. TTGs also have the ECS property, as well as games with superadditive valuations.

We now define how agents share their payoffs. The *set of payoff vectors* $X^*(G)$ of a game $G = (N, v)$ is the set of all feasible payoff vectors for its crisp analogue:

$$X^*(G) = \{\mathbf{x} \in \mathbb{R}_+^n \mid \sum_{i=1}^n x^i \leq v^*(N)\}.$$

An *OCF solution concept* is a function that assigns every OCF game a subset of its payoff vectors. Note that the definition of a payoff vector allows transfers between different partial coalitions. However, usually we want to divide payoffs in a way that respects the coalition structure. The next definition paves the way for this.

DEFINITION 2.2. *An imputation for a coalition structure $CS = (\mathbf{c}_1, \dots, \mathbf{c}_k) \in CS_N$ is a $n \times k$ matrix $\mathbf{x} = (\mathbf{x}_1, \dots, \mathbf{x}_k) \in M_{n \times k}(\mathbb{R}_+)$ that satisfies:*

- *Individual Rationality:* $\sum_{j=1}^k x_j^i \geq v^*(\{i\})$ for all $i \in N$.
- *Payoff Distribution:* for all $1 \leq j \leq k$ we have $\sum_{i=1}^n x_j^i \leq v(\mathbf{c}_j)$, and if $c_j^i = 0$ then $x_j^i = 0$.

An imputation is a way for members of each partial coalition to divide profits among themselves; observe that inter-coalitional transfers are not allowed. We call a tuple (CS, \mathbf{x}) of a coalition structure and an imputation a *feasible outcome*, let $I(CS)$ denote the set of all matrices \mathbf{x} such that (CS, \mathbf{x}) is a feasible outcome, and let $\mathcal{F}(S)$ denote the set of all feasible outcomes over S .

Let $(CS, \mathbf{x}) \in \mathcal{F}(N)$ be a feasible outcome. We define the *payoff* to an agent $i \in N$ as $p_i(CS, \mathbf{x}) = \sum_{j=1}^k x_j^i$. This is the total payoff of i from all coalitions in CS . Similarly, the total payoff to a set J is $p_J(CS, \mathbf{x}) = \sum_{i \in J} p_i(CS, \mathbf{x})$. Note that the vector $(p_1(CS, \mathbf{x}), \dots, p_n(CS, \mathbf{x}))$ is a payoff vector for G , since $p_N(CS, \mathbf{x}) = \sum_{i=1}^n p_i(CS, \mathbf{x}) = \sum_{i=1}^n \sum_{j=1}^k x_j^i = \sum_{j=1}^k \sum_{i=1}^n x_j^i \leq \sum_{j=1}^k v(\mathbf{c}_j) = v(CS) \leq v^*(N)$.

The *support* of a coalition $\mathbf{c} \in [0, 1]^n$ is the set of all players who devote their resources to \mathbf{c} . They are "interested parties" that may be hurt by any change to \mathbf{c} ; we write $\text{supp}(\mathbf{c}) = \{i \in N \mid c^i > 0\}$.

Given a coalition structure $CS = (\mathbf{c}_1, \dots, \mathbf{c}_k)$ and some $M \subseteq \{1, \dots, k\}$, the coalitions whose indices are in M form a coalition structure; this coalition structure is denoted $R(CS, M)$.

Given a set $J \subseteq N$, we denote $K_J = \{j \in \{1, \dots, k\} \mid \text{supp}(\mathbf{c}_j) \subseteq J\}$.

DEFINITION 2.3. *The coalition structure CS reduced to J is defined as*

$$CS|_J = R(CS, K_J).$$

These are all coalitions that are supported only by members of J . We let $\overline{CS|_J}$ denote the complement of $CS|_J$ in CS ; $\overline{CS|_J}$ is the coalition structure consisting of all coalitions in CS that have non- J members in their support.

DEFINITION 2.4. *A coalition structure CS' is a deviation of J from CS if:*

- (1) $\overline{CS|_J} = (\mathbf{c}_1 \dots \mathbf{c}_m)$, $\overline{CS'|_J} = (\mathbf{d}_1 \dots \mathbf{d}_m)$ and there is some permutation of m elements, σ , such that for all $1 \leq l \leq m$:

$$\forall i \notin J : d_l^i = c_{\sigma(l)}^i \text{ and } \forall i \in J : d_l^i \leq c_{\sigma(l)}^i$$

- (2) $w(CS'|_J) = w(CS|_J) + \sum_{l=1}^m (\mathbf{c}_{\sigma(l)} - \mathbf{d}_l)$.

Note that if we assume that CS' is efficient, then condition (1) implies condition (2). The deviation CS' describes how a set J retracts resources from some coalitions and uses them in order to maximize its own welfare. Given a deviation CS' of J from CS , we define $v^*(CS, J, CS')$ to be $v^*(w(CS'|_J))$; if J withdraws all of its resources from $\overline{CS|_J}$, then the total weight available to J is e^J , and $v^*(CS, J, CS') = v^*(J)$. For brevity, given some \mathbf{c}_l in $\overline{CS|_J}$ that J deviated from, we refer to the coalition after the deviation as $\text{dev}_{CS'}(\mathbf{c}_l)$.

3. THE ARBITRATION FUNCTION

To discuss stability in OCF games, we need to describe how agents react if some J deviates from (CS, \mathbf{x}) . Paper [4] presents three different alternatives for such a reaction. The *conservative deviation*, or c-deviation, completely denies payoffs to J , even from coalitions that J did not affect. A relaxation of this approach leads to the notion of a *refined deviation*, or r-deviation. Under this deviation rule, deviators receive their share of the profit from all coalitions that were unaffected by the deviation, i.e. if no member of the deviating subset J changed his contribution to a coalition \mathbf{c} , then J 's payoff from \mathbf{c} is the same as before the deviation. Under the *optimistic deviation*, or o-deviation, the players in J receive their share of the profits from any coalition in which all non-deviators can still earn the same payoff as before the deviation. Paper [4] defines three notions of the core that correspond to the deviations.

One could easily think of many other reactions to deviation; J receives only half of its original payoffs in all coalitions outside of J 's support, J receives payoff only from those agents who are not worse off after the deviation, and many others. Note that all such rules can be thought of as a payoff function that is given the original coalition structure and J 's deviation, and then decides on an appropriate payoff to J . We call this function an *arbitration function* or an *arbitrator*. This function decides how much J gets from each coalition, given the nature of its deviation.

3.1 Arbitration Functions

Suppose that we are given an outcome $(CS, \mathbf{x}) \in \mathcal{F}(N)$, a set of agents $J \subseteq N$ and a deviation CS' of J from CS . Set $\overline{CS|_J} = (\mathbf{c}_1 \dots \mathbf{c}_m)$ and $\overline{CS'|_J} = (\mathbf{d}_1 \dots \mathbf{d}_m)$.

DEFINITION 3.1. *The arbitration function is a mapping that assigns a real value to each coalition in $\overline{CS|_J}$.*

$$A(CS, \mathbf{x}, J, CS') = (\phi_l(CS, \mathbf{x}, J, CS'))_{l=1}^m$$

where ϕ_l is a function that determines how much the coalition \mathbf{c}_l is willing to give J given its deviation. We require ϕ_l to satisfy the following constraints:

- (1) ϕ_l is non-negative.
- (2) $\phi_l(CS, \mathbf{x}, J, CS')$ is only distributable between $J \cap \text{supp}(\mathbf{c}_l)$.
- (3) If $v(\text{dev}_{CS'}(\mathbf{c}_l)) - p_{N \setminus J}(\mathbf{c}_l, \mathbf{x}) \geq 0$, then $\phi_l(CS, \mathbf{x}, J, CS') \leq v(\text{dev}_{CS'}(\mathbf{c}_l)) - p_{N \setminus J}(\mathbf{c}_l, \mathbf{x})$, otherwise $\phi_l(CS, \mathbf{x}, CS') = 0$.
- (4) ϕ_l is deviation-monotone: If $K \subseteq J$ withdraws less resources from \mathbf{c}_l , then its payoff from ϕ_l is higher.

Condition (2) means that members of J that were not entitled to payoffs from \mathbf{c}_l before the deviation are not entitled to payoffs after deviating; condition (3) states that a deviating set J must ensure that all members of $N \setminus J$ in the support of \mathbf{c}_l are paid what they received under \mathbf{x} if it hopes to receive any profit from \mathbf{c}_l ; condition

(4) guarantees that the arbitrator behaves in a reasonable manner: agents know that their payoff from c_l is inversely proportional to the degree to which they hurt c_l . We illustrate the logic behind the concept of an arbitrator in the following example.

EXAMPLE 3.2. Consider a two-player TTG where there are three tasks: t_1, t_2, t_3 with $w(t_1) = 5, w(t_2) = 3, w(t_3) = 2$ and $p(t_1) = 10, p(t_2) = 7, p(t_3) = 4$. We have two agents: Alice, who has weight 4, and Bob, who has weight 1. The total weight of Alice and Bob is 5; they can complete t_2 and t_3 together and earn a total of 11. However, for the sake of our discussion, suppose that Alice and Bob agree on completing t_1 using all of their weight, and dividing the payoff between them so that Alice receives 6 and Bob receives 4. The corresponding outcome is: $((\frac{1}{1}), (\frac{6}{4}))$. Consider the following deviation by Alice; she withdraws 2 units of weight in order to complete t_3 on her own. This means that she still contributes 1 unit of weight to working with Bob, and they can still complete t_2 , earning 7. According to condition (3) Bob must receive at least what he did before the deviation. Therefore, the most that Alice can expect to get from t_2 is 3, under any arbitrator.

REMARK 3.3. We can allow the arbitrator more flexibility by permitting it to collect fines/make additional payments to the deviating subset. This can be captured by adding a *freely distributable value*, or a function $\Psi(CS, \mathbf{x}, J, CS')$ to the arbitrator, where Ψ is also deviation-monotone. This model is more general as Ψ can be distributed among *all* members of J in any way they wish, while ϕ_l may only be distributed among the members of $J \cap \text{supp}(c_l)$. Ψ can be, for example, a constant arbitration fee that must be paid by a deviating set. This fee is not related to any specific coalition, so the cost can be distributed between the agents in J in any way they see fit. All proofs in our paper go through for this more general model.

DEFINITION 3.4. The arbitration value of \mathcal{A} is

$$v^*(CS, J, CS') + \sum_{l=1}^m \phi_l(CS, \mathbf{x}, J, CS')$$

and denoted $\text{val}(\mathcal{A}, CS, \mathbf{x}, J, CS')$.

The arbitration value is the total payoff to a deviating set J given its deviation. The payoff is comprised of the most that J can make on its own plus the total payoff J receives from the coalitions it formed with non- J members; the greater the arbitration value, the higher the incentive to deviate.

3.2 The Arbitrated Core

In the spirit of the definition given in [3], we now define a profitable deviation of a subset in an arbitrated game.

DEFINITION 3.5. Let \mathcal{A} be an arbitrator over G . An \mathcal{A} -profitable deviation of J from an outcome (CS, \mathbf{x}) is an outcome (CS', \mathbf{y}) , where

1. CS' is a deviation of J from CS .
2. For all c_l in $\overline{CS|_J}$, $p_J(\{dev_{CS'}(c_l)\}, \mathbf{y}) \leq \phi_l(CS, \mathbf{x}, CS')$.
3. For all c_l in $\overline{CS|_J}$, if $\phi_l(CS, \mathbf{x}, CS') > 0$, then for all $i \in N \setminus J$, the payoff to i from the coalition $dev_{CS'}(c_l)$ is equal to her payoff under c_l .
4. $v(CS'|_J) = v^*(CS, J, CS')$ and \mathbf{y} reduced to the coalitions in $CS'|_J$ is an imputation over J .
5. For any $j \in J$, $p_j(CS', \mathbf{y}) > p_j(CS, \mathbf{x})$.

Condition 3 implies that a coalition c_l agrees to pay a deviating J only if each non- J member gets the same payoff it received under \mathbf{x} .

DEFINITION 3.6. The \mathcal{A} -core of $G = (N, v)$ is the set of all feasible outcomes in $\mathcal{F}(N)$ that no subset of agents has an \mathcal{A} -profitable deviation from. The \mathcal{A} -core is denoted $\mathcal{C}(\mathcal{A}, G)$.

EXAMPLE 3.7. The arbitration function for the c -core is $\phi_l \equiv 0$. The arbitration function for the r -core is $p_J(c_l, \mathbf{x})$ if c_l is the same after the deviation and is 0 otherwise. The arbitration function for the o -core is $\max\{0, v(dev_{CS'}(c_l)) - p_{N \setminus J}(c_l, \mathbf{x})\}$. We can define other forms of arbitrators. Set

$$N' = \{i \in N : i \notin J \text{ and } \exists c_l, i \in \text{supp}(c_l), dev_{CS'}(c_l) \neq c_l\}.$$

N' is the set of all non-members of J who were hurt in some way by J 's deviation. One could naturally assume that players in N' would not like to pay members of J anymore in any coalition. In this case, the arbitration function would be $p_J(c_l, \mathbf{x})$ if $\text{supp}(c_l) \cap N' = \emptyset$ and 0 otherwise. We denote the core that corresponds to this arbitrator the *sensitive core*.

Note that if J can \mathcal{A}_1 -profitably deviate using some deviation CS' and $\mathcal{A}_1(CS, \mathbf{x}, J, CS') \leq \mathcal{A}_2(CS, \mathbf{x}, J, CS')$ (coordinate-wise), then J can \mathcal{A}_2 -profitably deviate from (CS, \mathbf{x}) . This implies that if for all outcomes (CS, \mathbf{x}) and all deviations CS' of any $J \subseteq N$ we have $\mathcal{A}_1(CS, \mathbf{x}, J, CS') \leq \mathcal{A}_2(CS, \mathbf{x}, J, CS')$, then $\mathcal{C}(\mathcal{A}_2, G) \subseteq \mathcal{C}(\mathcal{A}_1, G)$. Particularly we have

$$o\text{-core} \subseteq r\text{-core} \subseteq \text{sensitive-core} \subseteq c\text{-core}$$

This is a generalization of the result shown in [4].

4. CHARACTERIZATION OF THE ARBITRATED CORE

We now give a general characterization of the core under some arbitration function \mathcal{A} . Our proof method is similar to the proof of the characterization result given in [4].

THEOREM 4.1. If $G = (N, v)$ has the ECS property, then an outcome $(CS, \mathbf{x}) \in \mathcal{F}(N)$ is in $\mathcal{C}(\mathcal{A}, G)$ if and only if for any $J \subseteq N$ and deviation CS' we have $p_J(CS, \mathbf{x}) \geq \text{val}(\mathcal{A}, CS, \mathbf{x}, J, CS')$.

Simply put, an outcome is stable if and only if for any coalition J and any deviation proposed by J , the payoff that the members of J can obtain under \mathcal{A} does not exceed their current payoff.

PROOF. Suppose first that for every $J \subseteq N$ and every deviation CS' of J from CS we have $p_J(CS, \mathbf{x}) \geq \text{val}(\mathcal{A}, CS, \mathbf{x}, J, CS')$. Therefore, for all $\mathbf{y} \in I(CS')$,

$$p_J(CS', \mathbf{y}) \leq \text{val}(\mathcal{A}, CS, \mathbf{x}, J, CS') \leq p_J(CS, \mathbf{x}).$$

Hence, there exists a player $j \in J$ that does not strictly benefit from (CS', \mathbf{y}) , and J cannot \mathcal{A} -profitably deviate from (CS, \mathbf{x}) .

Conversely, suppose that for some nonempty $J \subseteq N$ there exists a deviation CS' such that $p_J(CS, \mathbf{x}) < \text{val}(\mathcal{A}, CS, \mathbf{x}, J, CS')$. We show that (CS, \mathbf{x}) is not in the \mathcal{A} -core of G . Let $\overline{CS|_J} = (c_1 \dots c_m)$. For all $j \in J$ let $p_j = p_j(CS, \mathbf{x})$ and for all c_l in $\overline{CS|_J}$, let $r_l = \phi_l(CS, \mathbf{x}, J, CS')$. As v has the ECS property, $v^*(CS, J, CS') = v(CS_M)$ for some coalition structure $CS_M \in CS_J$.

$\text{val}(\mathcal{A}, CS, \mathbf{x}, J, CS') = v(CS_M) + \sum_{l=1}^m r_l$, which is strictly greater than $p_J(CS, \mathbf{x})$; while J can strictly gain by deviating, it is possible that the members of J cannot divide the payoffs from the deviation in a manner that strictly benefits all of them. We now

show that there is a subset of J that can profitably deviate. Recall that r_l may only be distributed among $J_l = \text{supp}(\mathbf{c}_l) \cap J$. Given r_l , we define the set of all viable payoff divisions of r_l among the members of J_l as

$$\Delta_l = \{\rho \in \mathbb{R}_+^n \mid \sum_{i=1}^n \rho^i = r_l \text{ and } \rho^i = 0 \text{ for all } i \notin J_l\}.$$

Note that Δ_l is compact. Given (CS, \mathbf{x}) , we define its *total loss function*

$$TL_{(CS, \mathbf{x})} : I(CS_M) \times \prod_{l=1}^m \Delta_l \rightarrow \mathbb{R}.$$

Given an imputation $\mathbf{y} \in I(CS_M)$ and $(\rho_l)_{l=1}^m \in \prod_{l=1}^m \Delta_l$, we define the total payoff to player $j \in J$ as

$$q_j = p_j(CS_M, \mathbf{y}) + \sum_{l=1}^m \rho_l^j.$$

The total loss of a payoff division is

$$TL_{(CS, \mathbf{x})}(\mathbf{y}, (\rho_l)_{l=1}^m) = \sum_{j \in J \mid p_j > q_j} p_j - q_j.$$

$TL_{(CS, \mathbf{x})}$ is a continuous, real valued function over a compact set, so there is some payoff division $(\mathbf{y}, (\rho_l)_{l=1}^m)$ that minimizes $TL_{(CS, \mathbf{x})}$. Given a loss-minimizing payoff division $(\mathbf{y}, (\rho_l)_{l=1}^m)$, we construct a directed graph $\Gamma = (V, E)$ where $V = J$, and there is a directed edge from $i \in J$ to $j \in J$ if and only if i can legally transfer payments to j ; this can happen if and only if both i and j are in the support of some coalition \mathbf{c} and i receives a positive payoff from \mathbf{c} . We color the vertices of the graph as follows: a vertex j is green if $p_j < q_j$, white if $p_j = q_j$, and red if $p_j > q_j$. Since $\sum_{j \in J} q_j > p_J(CS, \mathbf{x})$, the graph has at least one green vertex. If all vertices are green, then $p_j < q_j$ for all $j \in J$, i.e., CS' is a profitable deviation for all players in J , and we are done. We now assume that there is at least one non-green vertex.

Note that if $g \in J$ is green, then if there is an edge from g to some j , then g can transfer a small amount of payoff $0 < \delta < q_g - p_g$ to j . If δ is small enough, then the resulting outcome is still a viable imputation. Similarly, j can legally transfer the same amount to any vertex that j is connected to. Therefore, if there is a path from a green vertex g to some j , then g can transfer a small amount δ to j while remaining green. Following [4], we observe that since we chose a payoff distribution that minimizes $TL_{(CS, \mathbf{x})}$, if there is a path from a green vertex g to some vertex i , then i is not red. Thus, we can assume w.l.o.g. that if a vertex i is not green, then there is no path from a green vertex to i . Let G_J be the set of all green vertices; we claim that G_J can \mathcal{A} -profitably deviate. Indeed, note that making $J \setminus G_J$ return to their original contributions according to CS will not negatively affect G_J (here we use the fact that \mathcal{A} is deviation-monotone); if G_J decides to deviate from CS , without having $J \setminus G_J$ deviate as well, its payoffs cannot decrease. Therefore, G_J can \mathcal{A} -profitably deviate from CS , and we are done. \square

4.1 The Refined and Optimistic Cores

Theorem 4.1 immediately implies the characterization of the conservative core given in [4]. Let \mathcal{A}^c denote the conservative arbitrator; under \mathcal{A}^c we have $\phi_l(CS, \mathbf{x}, J, CS') \equiv 0$, so any deviating set should not leave any of its resources in any coalition, but rather devote all of its resources to maximize $v^*(CS, J, CS')$. Therefore,

$$\sup_{CS'} \{val(\mathcal{A}^c, CS, \mathbf{x}, J, CS')\} = \sup_{CS'} \{v^*(CS, J, CS')\} = v^*(J).$$

Our characterization result indeed shows that (CS, \mathbf{x}) is c-stable if and only if for all $J \subseteq N$, $p_J(CS, \mathbf{x}) \geq v^*(J)$.

Theorem 4.1 also gives an intuitive characterization of the refined and optimistic cores; under the refined arbitrator, denoted \mathcal{A}^r , a deviating subset J can expect payoff only from coalitions it did not change. Thus, if J decides to deviate from a coalition, it should withdraw all of its resources from that coalition. The arbitration value of \mathcal{A}^r is $v^*(CS, J, CS') + p_J(U, \mathbf{x})$, where U is a matrix whose columns are the coalitions unchanged by J 's deviation. Consequently, an outcome (CS, \mathbf{x}) is in the r-core if and only if for all $J \subseteq N$ and any deviation of J , CS' , we have $p_J(CS, \mathbf{x}) \geq v^*(CS, J, CS') + p_J(U, \mathbf{x})$. To conclude, an outcome (CS, \mathbf{x}) is in the r-core if and only if for any $J \subseteq N$ and any matrix Q whose column vectors are coalitions in $\overline{CS|_J}$ we have

$$p_J(CS|_J, \mathbf{x}) + p_J(Q, \mathbf{x}) \geq v^*(w(CS|_J) + w(Q)|_J).$$

Note that if $Q = \overline{CS|_J}$, we get the c-core condition.

Under the optimistic arbitrator, denoted \mathcal{A}^o , J can expect payoff from a coalition \mathbf{c} if all non- J members of \mathbf{c} get the payoff they received under \mathbf{x} . Thus, the payoff available to J is $v(\text{dev}_{CS'}(\mathbf{c}_l)) - p_{N \setminus J}(\mathbf{c}_l, \mathbf{x})$. Let us denote by $P(CS)$ the coalitions from which J can expect payoff if it makes the deviation CS' , and by $P(CS')$ the same coalitions after J 's deviation. We define $N(CS)$ to be the coalitions that will not pay J . By Theorem 4.1 we have that $(CS, \mathbf{x}) \in \mathcal{C}(\mathcal{A}^o, G)$ if and only if

$$p_J(CS, \mathbf{x}) \geq v^*(CS, J, CS') + v(P(CS')) - p_{N \setminus J}(P(CS), \mathbf{x}).$$

Since $p_N(P(CS), \mathbf{x}) = v(P(CS))$, it follows that $(CS, \mathbf{x}) \in \mathcal{C}(\mathcal{A}^o, G)$ if and only if

$$p_J(N(CS), \mathbf{x}) \geq v^*(CS, J, CS') + v(P(CS')) - v(P(CS)).$$

Note that if we only consider deviations where J withdraws all of its resources from $N(CS)$ and does not change its contribution to $P(CS)$, then we get the characterization of the r-core. Also note that checking if an outcome is in the r-core or c-core can be done by considering a finite number of deviations, but for o-core this is not the case.

5. THE NUCLEOLUS OF AN ARBITRATED OCF GAME

Although the core of a game is a useful solution concept, it may be empty in some cases; it is desirable to have a solution concept that is more robust, and, in particular, is guaranteed to be non-empty for all (reasonable) OCF games. In the non-overlapping setting, this role is fulfilled by the nucleolus [11]. We extend the notion of nucleolus to OCF games, and show that it exhibits many of the desirable properties of its non-OCF counterpart.

Let $\mathcal{A}^*(CS, \mathbf{x}, J)$ denote the most that $J \subseteq N$ can receive from an arbitrator given an outcome (CS, \mathbf{x}) . In this section, we only consider OCF games for which $\mathcal{A}^*(CS, \mathbf{x}, J)$ is a well-defined real value for any CS, \mathbf{x} and J .

DEFINITION 5.1. *Given an outcome (CS, \mathbf{x}) , the excess of $J \subseteq N$ is defined as $e(CS, \mathbf{x}, J) = \mathcal{A}^*(CS, \mathbf{x}, J) - p_J(CS, \mathbf{x})$.*

The excess is a measure of a subset's "unhappiness" with a given outcome: the lower the excess, the happier the subset. Note also that Theorem 4.1 states that $(CS, \mathbf{x}) \in \mathcal{C}(\mathcal{A}, G)$ if and only if $e(CS, \mathbf{x}, J) \leq 0$ for all $J \subseteq N$. Given an outcome (CS, \mathbf{x}) , we define its *excess vector* as

$$\theta(CS, \mathbf{x}) = (e(CS, \mathbf{x}, S_1), e(CS, \mathbf{x}, S_2), \dots, e(CS, \mathbf{x}, S_{2^n})),$$

where $e(CS, \mathbf{x}, S_1) \geq \dots \geq e(CS, \mathbf{x}, S_{2^n})$. We write $(CS, \mathbf{x}) \preceq_L (CS', \mathbf{y})$ if $\theta(CS, \mathbf{x})$ is lexicographically smaller than $\theta(CS', \mathbf{y})$.

We point out that Definition 5.1 coincides with the definition of excess for classic TU cooperative games; given a classic TU game (N, u) , the most that the set J can get is simply $u(J)$, and the excess is defined as the difference between $u(J)$ and the payoff to J . This analogous definition gives rise to an analogous definition of an arbitrated nucleolus.

Given an OCF game $G = (N, v)$ arbitrated by \mathcal{A} , the arbitrated nucleolus of G , denoted $\mathcal{N}(\mathcal{A}, G)$, is the set of all outcomes in $\mathcal{F}(N)$ that are minimal with respect to \preceq_L . Observe that just like in the non-overlapping case, if $\mathcal{C}(\mathcal{A}, G) \neq \emptyset$, then $\mathcal{N}(\mathcal{A}, G) \subseteq \mathcal{C}(\mathcal{A}, G)$.

5.1 Non-Emptiness of the Nucleolus

Unlike the arbitrated core, the nucleolus is never empty as long as $\mathcal{A}^*(CS, \mathbf{x}, J)$ is continuous with respect to (CS, \mathbf{x}) . In fact, it suffices that the excess of a set be achievable by using some deviation from a given outcome. This is true for the arbitrators defined above, assuming that v has the ECS property.

THEOREM 5.2. *If v has the ECS property and $\mathcal{A}^*(CS, \mathbf{x}, J)$ is continuous w.r.t (CS, \mathbf{x}) , then $\mathcal{N}(\mathcal{A}, G) \neq \emptyset$*

PROOF. First, we would like to note that the excess vector is comprised of continuous functions over $\mathcal{F}(N)$. Indeed, observe that for any outcome (CS, \mathbf{x}) and any $k = 1, \dots, 2^n$ we have

$$\theta_k(CS, \mathbf{x}) = \max_{S_1, \dots, S_k \subseteq N} \{\min\{e(CS, \mathbf{x}, S_1), \dots, e(CS, \mathbf{x}, S_k)\}\},$$

where all S_1, \dots, S_k are different subsets of N . Since $\mathcal{A}^*(CS, \mathbf{x}, J)$ is continuous, so is the excess. Thus, θ_k is obtained by combining continuous functions using a finite number of min and max operations, and therefore it is continuous as well.

Set $X_1 = \{(CS, \mathbf{x}) = \operatorname{argmin}_{(CS', \mathbf{y}) \in \mathcal{F}(N)} \{\theta_1(CS', \mathbf{y})\}\}$, and for every $k = 2, \dots, 2^n$, let

$$X_k = \{(CS, \mathbf{x}) = \operatorname{argmin}_{(CS', \mathbf{y}) \in X_{k-1}} \{\theta_k(CS', \mathbf{y})\}\}.$$

$X_{2^n} \subseteq \mathcal{N}(\mathcal{A}, G)$, since if $(CS, \mathbf{x}) \in X_k$ then $\theta_k(CS, \mathbf{x}) \leq \theta_k(CS', \mathbf{y})$ for every $k = 1, \dots, 2^n$. Thus, it remains to show that X_{2^n} is non-empty. Now, the set $\mathcal{F}(N)$ is compact and non-empty. From elementary calculus, we know that if $C \subseteq \mathbb{R}^m$ is a non-empty compact set, and $f : C \rightarrow \mathbb{R}$ is a continuous function, then the set $X = \{x \in C \mid f(x) = \min_{y \in C} \{f(y)\}\}$ is a non-empty compact set. Hence, X_1 is compact and non-empty, and inductively so is X_{2^n} . Consequently, $\mathcal{N}(\mathcal{A}, G) \neq \emptyset$. \square

5.2 Properties of the Nucleolus

The nucleolus in the non-overlapping setting exhibits some attractive properties. For example, in the non-overlapping setting, the nucleolus is a single point [9, 18]. In the arbitrated OCF setting, however, the nucleolus may have a richer structure.

EXAMPLE 5.3. Consider the following TTG: $N = \{1, 2\}$. Both players have weight of 1 and there is one task t with $w(t) = 2, p(t) = 20$. Assume that G is arbitrated by the refined arbitrator. The r -core of the game is not empty and the only coalition structure that is in the r -core is $CS = \binom{1}{1}$. Let us consider a payoff distribution where player 1 gets $10 - \varepsilon$ and player 2 gets $10 + \varepsilon$ where $0 < \varepsilon < 10$. The maximum value that can be provided to player 1 under the refined arbitrator is if he offers CS as his objection; any other deviation will leave him with nothing. Indeed, $\mathcal{A}^*(CS, \mathbf{x}, \{1\}) = p_1(CS, \mathbf{x}) = 10 - \varepsilon$, thus his excess is 0. One can verify that all nucleolus outcomes have excess of 0 for all sets

in this game. However, if the same game is arbitrated by the conservative arbitrator, then the excess of player 1 is $0 - (10 - \varepsilon) = \varepsilon - 10$, which will make him sensitive to the fact that he is being cheated.

Example 5.3 demonstrates that outcomes in $\mathcal{N}(\mathcal{A}, G)$ need not be unique, nor distribute payoff among players in the same manner. However, it turns out that if $\mathcal{A}^*(CS, \mathbf{x}, J)$ is convex as a function of \mathbf{x} when J and CS are fixed, then for any two outcomes in the nucleolus that have the same coalition structure, each subset of players has the same excess under both of these outcomes. First, we need the following technical lemma:

LEMMA 5.4. *If $(CS, \mathbf{x}), (CS, \mathbf{y}) \in \mathcal{N}(\mathcal{A}, G)$ and $\mathbf{z} = \frac{\mathbf{x} + \mathbf{y}}{2}$, then $(CS, \mathbf{z}) \in \mathcal{N}(\mathcal{A}, G)$.*

PROOF. Suppose that $(CS, \mathbf{x}), (CS, \mathbf{y}) \in \mathcal{N}(\mathcal{A}, G)$. Both outcomes must have the same excess vector, i.e. $\theta(CS, \mathbf{x}) = \theta(CS, \mathbf{y})$. Set $\mathbf{z} = \frac{\mathbf{x} + \mathbf{y}}{2}$. Since $I(CS)$ is convex, $\mathbf{z} \in I(CS)$. Consider $\theta(CS, \mathbf{z})$. Denote

$$\begin{aligned} \theta(CS, \mathbf{x}) &= (e(CS, \mathbf{x}, J_1), \dots, e(CS, \mathbf{x}, J_{2^n})), \\ \theta(CS, \mathbf{y}) &= (e(CS, \mathbf{y}, K_1), \dots, e(CS, \mathbf{y}, K_{2^n})), \\ \theta(CS, \mathbf{z}) &= (e(CS, \mathbf{z}, L_1), \dots, e(CS, \mathbf{z}, L_{2^n})). \end{aligned}$$

Given a deviation CS' of J from CS , we have

$$\mathcal{A}^*(CS, \mathbf{z}, J) \leq \frac{\mathcal{A}^*(CS, \mathbf{x}, J) + \mathcal{A}^*(CS, \mathbf{y}, J)}{2},$$

since \mathcal{A}^* is convex. Since the payoffs to J are linear in the imputations, we conclude that

$$e(CS, \mathbf{z}, J) \leq \frac{1}{2}e(CS, \mathbf{x}, J) + \frac{1}{2}e(CS, \mathbf{y}, J).$$

Denote $e(CS, \mathbf{x}, J_l) = e(CS, \mathbf{y}, K_l) = V_l$. We get

$$e(CS, \mathbf{z}, L_1) \leq \frac{e(CS, \mathbf{x}, L_1) + e(CS, \mathbf{y}, L_1)}{2} \leq V_1.$$

If at any point the inequality is strict, $e(CS, \mathbf{z}, L_1) < V_1$ and $\theta(CS, \mathbf{z})$ is strictly smaller lexicographically than $\theta(CS, \mathbf{x})$, a contradiction. We similarly conclude that $e(CS, \mathbf{z}, L_k) = V_k$ for all $k = 1, \dots, 2^n$. Therefore $\theta(CS, \mathbf{z}) = \theta(CS, \mathbf{x}) = \theta(CS, \mathbf{y})$, and $(CS, \mathbf{z}) \in \mathcal{N}(\mathcal{A}, G)$. \square

THEOREM 5.5. *Let $G = (N, v)$ be a game arbitrated by some convex arbitrator \mathcal{A} . If $(CS, \mathbf{x}), (CS, \mathbf{y}) \in \mathcal{N}(\mathcal{A}, G)$ then for any $J \subseteq N$ we have $e(CS, \mathbf{x}, J) = e(CS, \mathbf{y}, J)$.*

PROOF. The proof scheme is somewhat similar to the proof that the nucleolus for non-OCF games is unique [18, 9]. Let (CS, \mathbf{x}) and (CS, \mathbf{y}) be in $\mathcal{N}(\mathcal{A}, G)$. Set $\mathbf{z} = \frac{\mathbf{x} + \mathbf{y}}{2}$. Using the same notation as in Lemma 5.4, we know that $e(CS, \mathbf{x}, J_1)$ is equal to $e(CS, \mathbf{y}, K_1)$ and $e(CS, \mathbf{z}, L_1)$, so

$$e(CS, \mathbf{x}, J_1) + e(CS, \mathbf{y}, K_1) = 2e(CS, \mathbf{z}, L_1).$$

As shown in Lemma 5.4,

$$2e(CS, \mathbf{z}, L_1) \leq e(CS, \mathbf{x}, L_1) + e(CS, \mathbf{y}, L_1).$$

By definition of J_1 , $e(CS, \mathbf{x}, L_1) \leq e(CS, \mathbf{x}, J_1)$ and similarly, $e(CS, \mathbf{y}, L_1) \leq e(CS, \mathbf{y}, K_1)$. This implies that

$$e(CS, \mathbf{y}, K_1) = e(CS, \mathbf{y}, L_1) = e(CS, \mathbf{x}, J_1) = e(CS, \mathbf{x}, L_1).$$

We can swap L_1 with J_1 in the excess ordering of (CS, \mathbf{x}) without changing the excess vector. This can be done inductively for any L_k . We conclude that if $(CS, \mathbf{x}), (CS, \mathbf{y}) \in \mathcal{N}(\mathcal{A}, G)$ then all sets have the same excess in both outcomes. \square

The conservative arbitrator is constant at $v^*(J)$, and thus convex. This allows us to use Theorem 5.5 to show that for the conservative arbitration function any two outcomes in the nucleolus correspond to identical payoff vectors.

COROLLARY 5.6. *If G is arbitrated by the conservative arbitrator, then for any $(CS, \mathbf{x}), (CS, \mathbf{y}) \in \mathcal{N}(\mathcal{A}^c, G)$ and any $i \in N$, $p_i(CS, \mathbf{x}) = p_i(CS, \mathbf{y})$.*

PROOF. Consider two outcomes $(CS, \mathbf{x}), (CS, \mathbf{y}) \in \mathcal{N}(\mathcal{A}^c, G)$ and a player i . By Theorem 5.5, $\mathcal{A}^{c*}(CS, \mathbf{x}, J) - p_i(CS, \mathbf{x}) = \mathcal{A}^{c*}(CS, \mathbf{x}, J) - p_i(CS, \mathbf{y})$. On the other hand, $\mathcal{A}^{c*}(CS, \mathbf{x}, J) = \mathcal{A}^{c*}(CS, \mathbf{y}, J) = v^*(\{i\})$, so $p_i(CS, \mathbf{x})$ must equal $p_i(CS, \mathbf{y})$. \square

We remark that one can also show that the refined arbitrator is convex. However, as illustrated by Example 5.3, the conclusion of Corollary 5.6 does not hold for the refined arbitrator, since the value of $\mathcal{A}_z^*(CS, \mathbf{z}, J)$ may depend on the vector \mathbf{z} .

6. THE SHAPLEY VALUE OF OCF GAMES

Introduced by L.S. Shapley in [12], the Shapley value is a central solution concept in classic cooperative game theory. We offer two possible extensions of the Shapley value to OCF games; one assumes a fixed coalition structure and is somewhat similar to the Shapley value for coalition structures defined in [2], while the other takes into account the ability of sets to maximize their profits using coalition structures and is similar to the classic notion defined in [12]. We show that both values are unique with regard to specific sets of axioms¹. In this section, we denote the Shapley value for crisp games by sv .

Our first definition assumes that the coalition structure CS is given; it is possible that the agents have agreed on some division of labor, or one was assigned to them by a central authority. The following definition of a value provides an axiomatic method of assessing the contribution of each player to CS . Given a game $G = (N, v)$ and a coalition $\mathbf{c} \in [0, 1]^n$ we set forth the following axioms for a value $(\Phi_1(N, v, \mathbf{c}), \dots, \Phi_n(N, v, \mathbf{c}))$.

- (1) **Coalitional Efficiency:** $\sum_{i=1}^n \Phi_i(N, v, \mathbf{c}) = v(\mathbf{c})$.
- (2) **Symmetry:** Two players $i, j \in N$ are *OCF-symmetric* if for all $\mathbf{x} \in [0, 1]^n$, $v(\mathbf{x}) = v(\mathbf{x}_{i \sim j})$, where $\mathbf{x}_{i \sim j}$ is \mathbf{x} with the i -th and j -th coordinates exchanged. If i, j are OCF-symmetric, then $\Phi_i(N, v, \mathbf{c}) = \Phi_j(N, v, \mathbf{c})$.
- (3) **Dummy Player:** Set \mathbf{c}_{-i} to be \mathbf{c} with the i^{th} coordinate set to 0. If $v(\mathbf{c}_{-i}) = v(\mathbf{c})$ then $\Phi_i(N, v, \mathbf{c}) = 0$.
- (4) **Additivity:** $\Phi_i(N, v, \mathbf{c}) + \Phi_i(N, u, \mathbf{c}) = \Phi_i(N, u + v, \mathbf{c})$.

We define $\alpha_v : [0, 1]^n \times 2^N \rightarrow \mathbb{R}$ as $\alpha_v(\mathbf{c}, S) = v(\mathbf{c}|_S)$. Given a coalition \mathbf{c} , the *coalitional OCF Shapley value* of \mathbf{c} , denoted $SV_i(N, v, \mathbf{c})$, is $sv_i(\alpha_v(\mathbf{c}, \cdot))$. One can verify that

- $\alpha_v(\mathbf{c}, N) = v(\mathbf{c})$;
- if two players are OCF-symmetric then they are symmetric in $\alpha_v(\mathbf{c}, \cdot)$;
- if i is a dummy then $\alpha_v(\mathbf{c}, S \cup \{i\}) = v(\mathbf{c}_{S \cup \{i\}}) = v(\mathbf{c}_S) = \alpha_v(\mathbf{c}, S)$, hence i is dummy in $\alpha_v(\mathbf{c}, \cdot)$;
- $\alpha_u(\mathbf{c}, S) + \alpha_v(\mathbf{c}, S) = u(\mathbf{c}_S) + v(\mathbf{c}_S) = (u + v)(\mathbf{c}_S) = \alpha_{u+v}(\mathbf{c}, S)$.

¹For the axiomatization of the classic Shapley value, see [9] chapter 8, pp. 151-179, as well as the detailed review in [16]

This shows that the OCF properties described above naturally translate to their equivalents in non-OCF games. Hence, the coalitional OCF Shapley value satisfies properties (1)–(4). To show uniqueness, we use the following construction: given a function $u : 2^N \rightarrow \mathbb{R}$, define $v : [0, 1]^n \rightarrow \mathbb{R}$ by setting $v(\mathbf{x}) = u(S)$ if $\mathbf{x} = \mathbf{c}_S$ for some $S \subseteq N$ and $v(\mathbf{x}) = 0$ otherwise. Clearly, we have $\alpha_v(\mathbf{c}, S) = u(S)$ for any $S \subseteq N$. Therefore, uniqueness of the coalitional OCF Shapley value follows from the uniqueness of the classic Shapley value. The coalitional OCF Shapley value can be extended to coalition structures by setting $SV_i(N, v, CS) = \sum_{j=1}^k SV_i(N, v, \mathbf{c}_j)$, where $CS = (\mathbf{c}_1 \dots \mathbf{c}_k)$. It is immediate that $SV(N, v, CS)$ is efficient, i.e., the sum of the players' values is $v(CS)$, and the value of each coalition is distributed only among those who support it.

An alternative approach for measuring power does not assume a preexisting coalition structure, but rather measures the a-priori marginal contribution of a player, as all players try to maximize social welfare by forming coalition structures. Young [17] gives a characterization of the Shapley value using the notion of *strong monotonicity*; we use a similar notion for a value. We begin by setting forth the desirable axioms.

- (1) **Strong Monotonicity:** if for some $u, v : [0, 1]^n \rightarrow \mathbb{R}$ and some $i \in N$ we have $v^*(\mathbf{c}) - v^*(\mathbf{c}_{-i}) \geq u^*(\mathbf{c}) - u^*(\mathbf{c}_{-i})$ for all $\mathbf{c} \in [0, 1]^n$, then $\Phi_i(N, v) \geq \Phi_i(N, u)$.
- (2) **Symmetry:** A value Φ is *symmetric* if for any two symmetric players i, j : $\Phi_i(N, v) = \Phi_j(N, v)$.
- (3) **Efficiency:** $\sum_{i=1}^n \Phi_i(N, v) = v^*(N)$.

Axiom (1) states that if a player i has higher marginal contribution to $v^*(\mathbf{c})$ than to $u^*(\mathbf{c})$ for any \mathbf{c} , then her value in v should be higher; this is a generalization of strong monotonicity as defined in [17]. Also note that if two players are OCF-symmetric, then they are symmetric as players in the crisp analogue of the game. Finally, these notions are only well-defined assuming that the game G has the ECS property. We define the *OCF Shapley value*, denoted $SV^*(N, v)$, as

$$SV_i^*(N, v) = sv_i(N, v^*).$$

Strong monotonicity, efficiency and symmetry are inherited from their classic counterparts for $sv(N, v^*)$. Note also that the class of crisp analogues of OCF games corresponds to the class of superadditive games. Recall that a function $u : 2^N \rightarrow \mathbb{R}$ is called *superadditive* if for all disjoint $S, T \subseteq N$ it holds that $v(S) + v(T) \leq v(S \cup T)$. One can verify that the crisp analogue of any OCF game is superadditive. Moreover, given a superadditive $u : 2^N \rightarrow \mathbb{R}$, one can define the function $v : [0, 1]^n \rightarrow \mathbb{R}$ to be $v(e^S) = u(S)$ and 0 otherwise; for all $S \subseteq N$, $v^*(S) = u(S)$. Therefore, the uniqueness of the Shapley value for superadditive games implies its uniqueness for the class of crisp analogues, which in turn implies its uniqueness for OCF games.

The two notions of Shapley value for OCF games considered above do not, in general, coincide, even if the coalition structure for which we compute the coalitional OCF Shapley value is socially optimal.

EXAMPLE 6.1. Consider a 3-player TTG with $w_1 = 5, w_2 = 2, w_3 = 1$, and two tasks, t_1, t_2 , with $p(t_1) = 6, p(t_2) = 12$ and $w(t_1) = 4, w(t_2) = 8$. Let us compute $SV^*(N, v)$. When player 1 is first or second he has marginal contribution of 6. When he is last, his marginal contribution is 12. Therefore, $SV_1^*(N, v) = 8$, and, by efficiency and symmetry, $SV_2^*(N, v) = SV_3^*(N, v) = 2$. However, consider a coalition structure where players work on two

copies of t_1 , and each of them contributes half of his resources to each copy. Then any player has non-zero marginal contribution only if he is last, in which case he contributes 12. Therefore $SV_1(N, v, CS) = SV_2(N, v, CS) = SV_3(N, v, CS) = 4$.

In Example 6.1, player 1 can contribute significantly more than the other players, but his coalitional OCF Shapley value is equal to theirs in CS . This is because in the *specific* coalition structure CS , his marginal contribution is the same as his peers'; if any one of them leaves a coalition, the value of the remaining coalition structure becomes zero.

Note also that if two players in a TTG have $w_i = w_j$, then $SV_i^*(N, v) = SV_j^*(N, v)$. However, this is not necessarily true for the coalitional Shapley value.

EXAMPLE 6.2. Consider a 2-player TTG where both players have weight $w \geq 2$, and there are two tasks; $w(t_1) = 2w - 1$, $w(t_2) = 1$ and $p(t_1) = M$, $p(t_2) = x$, where $(2w - 1)x \leq M$. We form CS so that player 1 contributes all of her weight to t_1 , while player 2 contributes $w - 1$ to t_1 , and completes t_2 by herself. When player 1 is first, then $v(\binom{1}{0}) = x$, while under the current coalition structure, player 2 can gain $2x$ on her own. $SV_1(N, v, CS) = \frac{1}{2}(x + M + x - 2x) = \frac{M}{2}$, and by efficiency $SV_2(N, v, CS) = \frac{M+2x}{2}$.

Example 6.2 implies that the difference between $SV_i^*(N, v)$ and $SV_i(N, v, CS)$ can be arbitrarily large.

7. CONCLUSIONS AND FUTURE WORK

Our work shows significant similarity between concepts from classic cooperative game theory and their OCF counterparts. We can also generalize the notion of the bargaining set [6] to OCF games; the resulting notion shares many properties with the bargaining set in crisp games. We omit these results due to space constraints. Other solution concepts, such as the ε -core, can be described using the arbitration function itself, using the freely distributable component mentioned in Remark 3.3.

The arbitrated OCF model is far from being fully explored. We would like to point out a few promising directions for further research. First, it is shown in [13] that the Shapley value is in the core of a convex game. We would like to see if this result extends to the OCF setting. While [4] define convex OCF games and show that their c-core is not empty, it is not clear if there is an outcome in the c-core of a convex game such that each player is paid exactly her OCF Shapley value. In order to do so, we must find some coalition structure that corresponds to such a payoff scheme.

Another promising direction is exploring processes of overlapping coalition formation. While some work has been done on overlapping coalition formation algorithms [8], coalitional stability is yet to be fully explored. Our work assumes that an outcome is exogenously determined, and does not describe a process under which a stable outcome may arise. While [4] proposes a coalition formation procedure for convex OCF games, it is not clear how to extend it to the general case. A decentralized coalition formation algorithm, where agents repeatedly form and dissolve coalitions until a stable coalition structure and payoff division are agreed upon, would be useful in many multi-agent scenarios. The notion of arbitrators may play a significant role in such a process, as the arbitration function can effectively control the degree to which agents will be inclined to deviate.

Finally, we would like to investigate the algorithmic properties of the solution concepts defined in this paper. For example, it is clear that it is generally hard to compute the OCF Shapley value,

even when one can compute v^* in polynomial time. However, it would be interesting to identify natural classes of games where the Shapley value is tractable.

8. ACKNOWLEDGMENTS

This research was supported by National Research Foundation (Singapore) under grant 2009-08, by NTU SUG (Edith Elkind), and by SINGA graduate fellowship (Yair Zick).

9. REFERENCES

- [1] J. Aubin. Cooperative fuzzy games. *Mathematics of Operations Research* 6(1):1–13, 1981.
- [2] R. Aumann and J. Drèze. Cooperative games with coalition structures. *International Journal of Game Theory*, 3:217–237, 1974.
- [3] G. Chalkiadakis, E. Elkind, E. Markakis, and N. Jennings. Overlapping coalition formation. In *WINE'08*, pp. 307–321, 2008.
- [4] G. Chalkiadakis, E. Elkind, E. Markakis, M. Polukarov, and N. Jennings. Cooperative games with overlapping coalitions. *Journal of Artificial Intelligence Research* 39:179–216, 2010.
- [5] 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*, pp. 635–640, 2006.
- [6] M. Davis and M. Maschler. Existence of stable payoff configurations for cooperative games. *Bulletin of the American Mathematical Society* 69:106–108, 1963.
- [7] P. Dubey. On the uniqueness of the Shapley value. *International Journal of Game Theory* 4(3):131–139, 1975.
- [8] C. Lin and S. Hu. Multi-task overlapping coalition parallel formation algorithm. In *AAMAS'07*, pp. 1260–1262, 2007.
- [9] B. Peleg and P. Sudhölter. *Introduction to the Theory of Cooperative Games*, Springer, second edition, 2007.
- [10] T. Michalak, J. Sroka, T. Rahwan, M. Wooldridge, P. McBurney and N. R. Jennings. A distributed algorithm for anytime coalition structure generation. In *AAMAS'10*, pp. 1007-1014, 2010.
- [11] D. Schmeidler. The nucleolus of a characteristic function game. *SIAM Journal on Applied Mathematics* 17:1163–1170, 1969.
- [12] L. Shapley. A value for n -person games. In *Contributions to the Theory of Games, vol. 2*, Annals of Mathematics Studies 28 pp. 307–317, 1953.
- [13] L. Shapley. Cores of convex games. *International Journal of Game Theory*, 1:11–26; errata, *ibid.* 1 (1971/72), 199, 1971/72.
- [14] O. Shehory and S. Kraus. Formation of overlapping coalitions for precedence-ordered task-execution among autonomous agents. In *ICMAS-96*, pp. 330–337, 1996.
- [15] T. Shrot, Y. Aumann, S. Kraus. On agent types in coalition formation problems. In *AAMAS'10*, pp. 757–764, 2010.
- [16] E. Winter. The Shapley value. *Handbook of Game Theory with Economic Applications* 3:2025–2054, 2002.
- [17] H. Young. Monotonic solutions of cooperative games. *International Journal of Game Theory* 14(2):65–72, 1985.
- [18] S. Zamir, M. Maschler, and E. Solan. *Game Theory*. Magnes Press, Hebrew University of Jerusalem, Israel, 2008.

Learning the Demand Curve in Posted-Price Digital Goods Auctions

Meenal Chhabra
Rensselaer Polytechnic Inst.
Dept. of Computer Science
Troy, NY, USA
chhabm@cs.rpi.edu

Sanmay Das
Rensselaer Polytechnic Inst.
Dept. of Computer Science
Troy, NY, USA
sanmay@cs.rpi.edu

ABSTRACT

Online digital goods auctions are settings where a seller with an unlimited supply of goods (e.g. music or movie downloads) interacts with a stream of potential buyers. In the posted price setting, the seller makes a take-it-or-leave-it offer to each arriving buyer. We study the seller's revenue maximization problem in posted-price auctions of digital goods. We find that algorithms from the multi-armed bandit literature like UCB, which come with good regret bounds, can be slow to converge. We propose and study two alternatives: (1) a scheme based on using Gittins indices with priors that make appropriate use of domain knowledge; (2) a new learning algorithm, LLVD, that assumes a linear demand curve, and maintains a Beta prior over the free parameter using a moment-matching approximation. LLVD is not only (approximately) optimal for linear demand, but also learns fast and performs well when the linearity assumption is violated, for example in the cases of two natural valuation distributions, exponential and log-normal.

Categories and Subject Descriptors

J.4 [Social and Behavioral Sciences]: Economics

General Terms

Algorithms, Economics

Keywords

Electronic markets, Economically-motivated agents, Single agent learning

1. INTRODUCTION

Digital goods auctions are those where a seller with an unlimited supply of identical goods interacts with a population of buyers who desire one unit of that good [12, 11]. These are typically thought of as digital goods which can be produced at negligible cost, for example, rights to watch a movie broadcast, or to download an audio file.

Consider the problem faced by a company that has the rights to a piece of music, and wants to market it to consumers. There is some underlying valuation distribution on

Cite as: Learning the Demand Curve in Posted-Price Digital Goods Auctions, Meenal Chhabra and Sanmay Das, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 63-70.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

the potential population of buyers, reflecting how much each potential buyer values that piece. However, the seller is not aware of this distribution, and can only learn it through interaction with buyers. The seller's goal is to maximize her own revenue. While such problems have typically been dealt with by using a few discrete possible prices and estimating popularity, this has mostly been due to the transaction costs associated with regularly changing prices. Dynamic pricing mechanisms, on the other hand, are increasingly available to sellers, and it is now practical to consider strategies that change prices online [13]. The typical interaction will be that the user searches a music database for the piece, sees a price, and decides whether or not to buy.

In this kind of posted-price mechanism [15, 3], the seller offers a single price, and an arriving buyer has the option to either complete the purchase at that price, or not go through with it. If the seller knew the distribution of valuations, the pricing problem for revenue maximization would be simple to solve, yielding a single fixed price to be offered to all the buyers (under the assumption that the seller has no way of discriminating between buyers, or finding out their individual valuations). This distribution can also be thought of as the demand curve, because an arriving buyer will only buy if her valuation exceeds the posted price being offered.

Posted price mechanisms have also received attention in the context of limited supply auctions [4]. There has been work in economics on learning the demand curve in posted price auctions when the seller has a single unit of the item to sell [5], and also on learning the demand curve using buyers' bidding behavior in non-posted price settings [19].

Posted price auctions in which the seller must learn the demand curve are a natural application for the tools of dynamic programming and reinforcement learning because they exhibit a classic exploration-exploitation dilemma. The quoted price serves as both a profit-seeking mechanism (exploitation) as well as an information-gathering one (exploration). In the context of two-sided posted-price mechanisms in finance where a "market maker" offers to both buy and sell a security at some price, Das and Magdon-Ismael [7] use dynamic programming techniques to show that there are times when it is optimal to make significant losses in order to learn the valuation distribution more quickly. In digital goods auctions the seller does not make a loss, but may lose out on potentially higher revenue instead.

Given the exploration-exploitation dilemma inherent in the problem, it is natural that many of the algorithms analyzed for posted price selling with unknown demand have been based on the multi-armed bandit literature. Several of

these schemes have been shown to possess good properties in terms of asymptotic regret for the seller’s revenue maximization problem in the unlimited supply setting. Blum *et al* [3] discuss the application of Auer *et al*’s [2] EXP3 algorithm for the adversarial multi-armed bandit problem to posted price mechanisms, showing a worst-case adversarial bound. Kleinberg and Leighton [15] derive regret bounds for Auer *et al*’s [1] UCB1 bandit algorithm for i.i.d. settings in the posted price context. UCB1 is intended to minimize regret even in finite-horizon contexts, so we would expect it to perform relatively well. However, these algorithms rarely perform very well in terms of utility received in even simulated posted price auction settings – for example, in Conitzer and Garera’s comparison of EXP3 with gradient ascent and Bayesian methods [6], or even in different applications, as found by Vermorel and Mohri on an artificially generated dataset and a networking dataset [20]. Conitzer and Garera’s Bayesian methods are a relevant comparison to the algorithms we develop here, but they make a “correct prior” assumption, mostly focusing on learning when the model is known but the parameters unknown (for example, when the valuation distribution is uniform or exponential with known probabilities and a set of possible parameters with finite support for each type of distribution).

Contributions.

In this paper, we study the problem of revenue maximization in posted-price auctions of digital goods from the perspective of reinforcement learning and maximizing flow utility, rather than trying to achieve asymptotic regret bounds. We evaluate algorithms on simulated buying populations, with valuations distributed uniformly, exponentially, and log-normally. We find that regret-minimization algorithms from the multi-armed bandit literature are slow to learn in practice, and hence impractical, even for simple distributions of valuations in the buying population. We propose two alternatives: (1) a scheme based on Gittins indices that starts with different priors on the arms based on the knowledge that purchases at higher prices are less likely, and (2) a new reinforcement learning algorithm for the problem, called LLVD, that is based on a plausible linearity assumption on the structure of the demand curve. LLVD maintains a Beta distribution as the seller’s belief state, updating it using a moment-matching approximation. LLVD is (approximately) optimal when the linearity assumption holds, and empirically performs well for several families of valuation distributions that violate the linearity assumption.

2. THE POSTED PRICE MODEL

We start by introducing the model and assumptions that we will use. Buyers arrive in a stream, each with an i.i.d. valuation v of the good from an unknown underlying distribution f_V . f_V can have support on $[0, \infty)$. At each instant in time, the seller quotes a price $q_t \in [0, \infty)$, a potential buyer arrives with $v_t \sim f_V$, and chooses to buy if $v_t \geq q_t$ and not to buy otherwise. The seller has access to the history of her own pricing decisions, as well as the purchase decisions made by each arriving buyer. Her goal is to sequentially set q_t so as to maximize (discounted) expected total long-term revenue (we assume an infinite horizon model).

2.1 Learning the Demand Curve

For any given distribution of buyer valuations f_V , under the assumption that buyer valuations are I.I.D. draws from f_V at each point in time, there is a single optimal price q_{OPT} that maximizes the seller’s expected revenue. When f_V is unknown, there are several different possible design goals. In this work we seek to design an algorithm that maximizes flow utility, rather than an algorithm with the explicit goal of asymptotically correct or regret-bounded learning. Therefore, we focus on a dynamic programming approach that maximizes flow utility under a probabilistic model. This is a problem that falls within the domain of dynamic programming, reinforcement learning, and optimal experimentation, because the seller’s actions, corresponding to posted prices, have both a profit role (exploitation) and an informational role (exploration; conveying information about the true demand curve). The first problem with designing such a model is that the seller’s state space is itself a probability distribution over possible probability distributions (of valuations), so without restricting the space of possibilities it is difficult to get any traction. It is useful to consider a simple example.

“Linear” Demand.

Assume that buyer valuations are distributed uniformly on $[0, B]$. The probability of an arriving buyer choosing to buy at price q , $P(q)$ is $(B - q)/B$, or $1 - \gamma q$ where $\gamma = 1/B$. This entails a linear form for the probability of a sale at price q , so we refer to this (loosely) as the case of linear demand.

Now consider a particularly simple example. Suppose the seller knows with certainty that the demand function is either F , corresponding to γ_1 , or G , corresponding to γ_2 . Let α denote the probability the seller associates with demand function F . Then the state space is entirely parameterized by α . The expected discounted revenue is given by $\pi(\alpha_t) = \sum_{k=t}^{\infty} \delta^{k-t} (\alpha_k q_k P_F(q_k) + (1 - \alpha_k) q_k P_G(q_k))$.

A revenue maximizing policy is a mapping from α to q that maximizes π . The states $\alpha = 0$ and $\alpha = 1$ have no uncertainty associated with them, and the problem reduces to a simple maximization. When $\alpha = 1$, we maximize $\max_q \sum_{k=0}^{\infty} \delta^k (q P_F(q)) = \max_q \frac{q P_F(q)}{(1-\delta)}$.

For this example we assume $q \in [0, 1]$. So if the optimal q is theoretically greater than 1, the item is priced at 1. The function itself is increasing up to a maximum at $q = 1/2\gamma_1$, so the maximum within our domain $q \in [0, 1]$ is at $q = \min(1/2\gamma_1, 1)$ if $\alpha = 1$. Similarly if $\alpha = 0$, then the optimal price is $q = \min(1/2\gamma_2, 1)$.

For general α , the seller sets a price q (since we are discussing optimal actions in a situation that is not explicitly time dependent, we suppress any dependence on t) Depending on the action of an arriving buyer, the seller updates α . If the buyer buys, then $\alpha' = \frac{\alpha P_F(q)}{\alpha P_F(q) + (1-\alpha) P_G(q)}$. For our particular model, $\alpha' = \frac{\alpha - \gamma_1 \alpha q}{1 + ((\gamma_2 - \gamma_1) \alpha - \gamma_2) q}$. If the buyer does not buy, the state update is $\alpha'' = \frac{\alpha(1 - P_F(q))}{\alpha(1 - P_F(q)) + (1-\alpha)(1 - P_G(q))}$. Again, for our particular model, $\alpha'' = \frac{\gamma_1 \alpha}{(\gamma_1 - \gamma_2) \alpha + \gamma_2}$. This latter equation is of particular interest, since there is, surprisingly, no dependence on q .

The relevant probabilities of buying and not buying, given a (state, action) pair consisting of α and q are given by $\Pr(\text{Buy}|\alpha, q) = \alpha P_F(q) + (1-\alpha) P_G(q)$, and $\Pr(\neg\text{Buy}|\alpha, q) = \alpha(1 - P_F(q)) + (1 - \alpha)(1 - P_G(q))$.

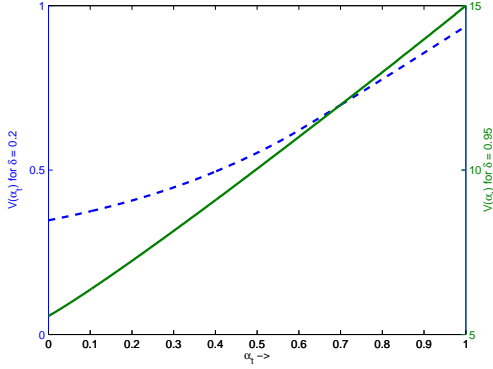


Figure 1: The value function for $\gamma_1 = 0.25, \gamma_2 = 0.9$ and discount factors $\delta = 0.2$ and $\delta = 0.95$. Note how the value function for high δ is almost linear.

Now we can write down the Bellman equation:

$$V(\alpha) = \alpha q P_F(q) + (1 - \alpha) q P_G(q) + \delta V' \quad (1)$$

where $V' = \Pr(\text{Buy}|\alpha, q)V(\alpha') + \Pr(\neg\text{Buy}|\alpha, q)V(\alpha'')$.

We now know the dynamics of the system. We can solve by discretizing α (we know $\alpha \in [0, 1]$) and using value iteration for any particular values of γ_1 and γ_2 . Figure 1 shows the value function for two different values of δ .

Computing the value function in this case leads to an interesting observation. When δ is high, the value function is almost linear in α . We can approximate the value function by $V = b\alpha + c$ to get an analytical approximate solution. Substituting in Equation 1 and finding b and c by equating coefficients, we find $V = \frac{Zq^2 + q}{1 - \delta}$ where $Z = (\gamma_2 - \gamma_1)\alpha - \gamma_2$. This equation implies that the optimal choice for q is the same as the myopically optimal choice! The linearity of the value function and the approximate optimality of a myopic strategy arise in part because, regardless of the strategy for setting q , good information is received by whether or not a buyer buys, allowing us to distinguish the populations, and α converges to either 0 or 1 quickly. This is partly a function of the fact that only one of the two possible future states α' and α'' depends in any way on q . In fact, the myopic approximation continues to be an excellent approximation to the optimal strategy even for lower values of δ , because at lower values immediate revenue dominates future revenue in the value function anyhow.

More General Settings.

The example discussed above is analytically tractable because of the restriction to two possible distributions, reducing our state space to a single continuous variable. This restriction is too onerous for any realistic application. The simplest way to remove this restriction without sending tractability overboard is to consider the whole space of linear demand functions with $\gamma \in [0, 1]$ (the restriction to $\gamma \leq 1$ is not restrictive, because the effect could be achieved through rescaling of the valuations). We approach this problem by maintaining a probability distribution over γ .

3. ALGORITHMS

Here we describe the three algorithms we compare for this problem: (1) our new parametric algorithm, LLVD; (2) a Gittins-index based strategy with appropriately chosen priors; (3) UCB, a regret-minimizing algorithm from the multi-armed bandit literature.

3.1 The LLVD Algorithm

Our main assumption is that it is reasonable to model the probability of an arriving buyer choosing to go through with a purchase at quoted price q as a linear function of q , $\Pr(\text{Buy}|q) = 1 - \gamma q$. This gives rise to our learning algorithm, which we call ‘‘Linear Learning of Valuation Distributions’’ (LLVD).

Under the linearity assumption we want to maximize total expected (discounted) revenue. The seller’s state space is now the space of distributions over γ . In order to make this a tractable state space to work with, we enforce that the seller always represents her beliefs as a Beta distribution ($\gamma \in [0, 1]$). The state space can then be parametrized by the two parameters of the Beta distribution. We need to derive the state space transition model and the reward model in order to solve for the seller’s optimal policy. In the following, $f(\gamma; \alpha, \beta)$ represents the density function for the Beta distribution. $F(\gamma; \alpha, \beta)$ represents the c.d.f for the Beta distribution, and $F_k(\gamma)$ represents $F(\gamma; \alpha + k, \beta)$.

Transition Model.

An arriving buyer is quoted a price q and decides whether or not to buy at that price. She will buy if her valuation is less than equal to the price quoted. The seller updates her own distribution over γ based on whether or not the arriving buyer bought the good. Consider the Bayesian updates in two cases:

1. Buyer does not buy:

$$\begin{aligned} f(\gamma|\neg\text{Buy}) &= \frac{f(\gamma; \alpha, \beta)(\gamma q)}{\int_0^{1/q} f(\gamma; \alpha, \beta)(\gamma q) d\gamma} = \frac{\gamma^\alpha (1 - \gamma)^{\beta-1}}{\int_0^{1/q} \gamma^\alpha (1 - \gamma)^{\beta-1} d\gamma} \\ &= \frac{f(\gamma; \alpha + 1, \beta)}{F(1/q, \alpha, \beta)} = \frac{f(\gamma; \alpha + 1, \beta)}{F_0(1/q)} \end{aligned}$$

For $q < 1$, the normalizing constant is 1 and the true posterior is Beta. When $q > 1$ the posterior need not be Beta, so we compute the Beta distribution that matches the first and second moment of the true posterior. This yields a pair of simultaneous equations for α_{t+1} and β_{t+1} (in the equations below F_k represents $F_k(1/q_t)$):

$$\begin{aligned} \frac{\alpha_{t+1}}{\alpha_{t+1} + \beta_{t+1}} &= \frac{q_t \mathbb{E}(\gamma^2) F_2 + \mathbb{E}(\gamma)(1 - F_1)}{(q_t \mathbb{E}(\gamma) F_1 + 1 - F_0)} \\ \frac{\alpha_{t+1}(\alpha_{t+1} + 1)}{(\alpha_{t+1} + \beta_{t+1})(\alpha_{t+1} + \beta_{t+1} + 1)} &= \frac{q_t \mathbb{E}(\gamma^3) F_3 + \mathbb{E}(\gamma^2)(1 - F_2)}{(q_t \mathbb{E}(\gamma) F_1 + 1 - F_0)} \end{aligned}$$

2. Buyer buys:

$$\begin{aligned} f(\gamma|\text{Buy}) &= \frac{f(\gamma; \alpha, \beta)(1 - \gamma q)}{\int_0^{1/q} f(\gamma; \alpha, \beta)(1 - \gamma q) d\gamma} \\ &= \frac{f(\gamma; \alpha, \beta)(1 - \gamma q)}{(F(1/q, \alpha, \beta) - q \mathbb{E}(\gamma) F(1/q, \alpha + 1, \beta))} \\ &= \frac{f(\gamma; \alpha, \beta)(1 - \gamma q)}{(F_0(1/q) - q \mathbb{E}(\gamma) F_1(1/q))} \end{aligned}$$

Again, we approximate the true posterior with a Beta distribution by matching the first and second moments.

$$\frac{\alpha_{t+1}}{\alpha_{t+1} + \beta_{t+1}} = \frac{\mathbb{E}(\gamma)F_1 - q_t F_2 \mathbb{E}(\gamma^2)}{F_0 - q_t \mathbb{E}(\gamma)F_1}$$

$$\frac{\alpha_{t+1}(\alpha_{t+1} + 1)}{(\alpha_{t+1} + \beta_{t+1})(\alpha_{t+1} + \beta_{t+1} + 1)} = \frac{\mathbb{E}(\gamma^2)F_2 - q_t \mathbb{E}(\gamma^3)F_3}{F_0 - q_t \mathbb{E}(\gamma)F_1}$$

Let M and S represent first and second order moments respectively. Solving these equations yields update rules $\alpha_{t+1} = \frac{MS - M^2}{M^2 - S}$ and $\beta_{t+1} = \frac{(1-M)\alpha_{t+1}}{M}$.

Reward Model.

Let π denote the discounted long-term revenue and δ the discount factor. Let $P(q) = \Pr(\text{Buy}|q)$. Then $\pi = q_0 P(q_0) + \sum_{t=1}^{\infty} q_t P(q_t)$. The first term, $\pi_0 = q_0 P(q_0)$ is the expected reward at this particular instant, from the next action. We can compute the expected value of this term:

$$P(q) = \int_0^{1/q} (1 - \gamma q) f(\gamma; \alpha, \beta) d\gamma$$

$$= F(1/q; \alpha, \beta) - q \mathbb{E}(\gamma) F(1/q; \alpha + 1, \beta)$$

$$= F(1/q; \alpha, \beta) - q \mu F(1/q; \alpha + 1, \beta) \quad (2)$$

where $\mu = \alpha/(\alpha + \beta)$.

$$\pi_0 = q_0 (F(1/q_0; \alpha, \beta) - q_0 \mathbb{E}(\gamma) F(1/q_0; \alpha + 1, \beta))$$

$$= q_0 (F(1/q_0; \alpha, \beta) - q_0 \mu F(1/q_0; \alpha + 1, \beta)) \quad (3)$$

The Bellman Equation.

In a risk-neutral framework, we can similarly take expectations over γ and derive the appropriate Bellman equation: $V(\alpha_t, \beta_t) = \max_q q P(q) + \delta V'$, where

$$V' = P(q)V(\alpha_{t+1}, \beta_{t+1}|\text{Buy}) + (1 - P(q))V(\alpha_{t+1}, \beta_{t+1}|\text{NoBuy})$$

Obviously, if γ were known to the seller, the optimal action would be the optimal myopic action, and it would yield a discounted expected revenue of:

$$\pi = \max_q (q(1 - \gamma q) + \sum_{t=1}^{\infty} \delta^t q(1 - \gamma q))$$

$$= \max_q \frac{q(1 - \gamma q)}{1 - \delta} = \max_q \frac{q(1 - \gamma q)}{1 - \delta} \quad (4)$$

This equation is maximized at $q = \frac{1}{2\gamma}$, in our environment, yielding $V = \frac{1}{4\gamma(1-\delta)}$.

Solving for the optimal policy.

Various issues arise in trying to solve such a system. A value-iteration type method would rely on a reasonable functional approximation of the value function in order to converge to a correct estimate. We use a different approach by first restricting the problem to a space where table-based value iteration can be applied, and then extrapolating to the complete space. We start by restricting to values of q between 0 and 1.

The $q < 1$ case: Equation 2 reduces to $P(q) = (1 - \mu q)$, therefore Equation 3 reduces to $\pi_0 = q_0(1 - \mu q_0)$ because $F(1/q) = 1$ for the Beta distribution as $q < 1$. Equation 4 is maximized at $q = \min(1, \frac{1}{2\mu})$, in our environment, yielding $V = \min(\frac{1}{4\mu(1-\delta)}, \frac{1-\mu}{1-\delta})$. Since the transition model is known

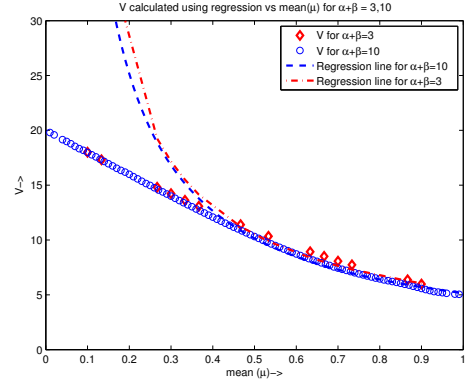


Figure 2: Comparison of the regression line with data from the value iteration table for different values of $\alpha + \beta$. Note the very tight match in the domain where the optimal q would be expected to be less than 1. The regression function allows LLVD to generalize this to the entire space (notice the difference between the line and the data points for lower values of μ , which correspond to higher optimal values of q).

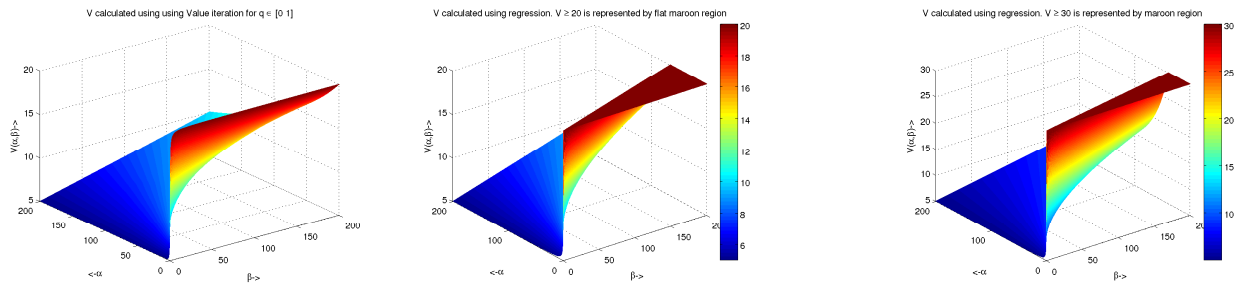
(the fact that the true posterior is Beta when the buyer buys for $q < 1$ is helpful in efficient implementation), all that remains in order to discretize and apply value iteration is to specify some boundary conditions on the model. The boundary conditions correspond to having a high degree of certainty about the value of γ . We assume that when the variance of the Beta distribution becomes less than 0.001, γ can be assumed to be known to the seller, and it is then equal to μ . In order for this technique to be consistent, we need to show that once the variance is sufficiently low, it will not be the case that it again starts increasing. We can show that in expectation the variance decreases in every iteration for $q < 1$; the proof is omitted due to space considerations.

This yields the final algorithm: we use value iteration to solve for the value function on a grid for $\alpha, \beta \in [0.1, 200]$, but we pre-fill all spaces where α, β are such that the variance of the distribution is less than 0.001. Figure 3 (V1) shows the value function for $\delta = 0.95$, as a function of α and β .

Extending to $q > 1$: We expect the value function computed using table-based value iteration to closely approximate the universally “correct” one for regions where the optimal value of q is less than 1. Therefore, we fit a regression line using values from the value function matrix where $\mu > 0.6$ (implying that the optimal q is probably lower than 0.85). Empirically, we find that the value function is close to linear in $\frac{1}{\mu}$ and $\frac{1}{\alpha + \beta}$ (see Figure 2). So we approximate the value function for the whole space as

$$V(\alpha, \beta) = a_1 \frac{\alpha + \beta}{\alpha} + a_2 \frac{1}{\alpha + \beta} \quad (5)$$

Figure 3 shows that this is a good approximation over the entire space. Now, at any time $T = t$, with the belief state



V1: Table-based value function V2: Value function using regression V3: Value function extrapolated using regression

Figure 3: V1 is the value function computed using table-based value iteration with $q < 1$ (the maximum value of V is 20). V2 is the value function computed using regression (see Equation 5), showing the similarity to V1 where the value function is less than 20 (the flat maroon region shows where $V \geq 20$, where the value functions would be expected to differ, and $q > 1$). V3 shows some more of the structure of the value function computed using regression (Equation 5) in the region where it attains values between 20 and 30.

(α, β) we can find the q which maximizes the given equation.

$$\pi = \max_{q_t} V(\alpha, \beta) + \delta(\Pr(\text{Buy}|q_t)V(\alpha', \beta'|\text{Buy}) + (1 - \Pr(\text{Buy}|q_t))V(\alpha'', \beta''|\neg\text{Buy}))$$

Here α' , β' , α'' and β'' are functions of q_t, α and β , price offered at time $T=t$. These values can be calculated as discussed above by comparing the first two moments.

Implementation notes: In our experiments, we compute the value function using $\delta = 0.95$. The best fit regression line is obtained for $a_1 = 4.99$ and $a_2 = 1.5147$; for convenience we use $a_1 = 5$ and $a_2 = 1.5$. The LLVD based seller then learns online, constantly updating her belief on γ (starting from $\alpha = \beta = 1$), and choosing the price that maximizes the value function at any instant.

3.2 Bandit Schemes

Multi-armed bandit algorithms are often applied to Dynamic pricing [16]. The different pricing options are the arms of the bandit and the goal is to find the arm that maximizes infinite horizon discounted reward. The downside of such approaches is that one needs to have fixed arms, and there is no “information sharing” between arms. How to discretize the space into arms is an interesting problem. For the purposes of this paper, we discretize the space from $[0.5, 2q^*]$ in 20 steps, where q^* is the (analytically computed) optimal price for the specific valuation distribution. While reasonable for evaluation, there may be situations where the need to find a reasonable interval is a downside for bandit-based methods. We discuss two algorithms.

A Gittins Index Scheme With Smart Priors.

Gittins and Jones introduced dynamic allocation indices as the Bayes optimal solution to the exploration-exploitation dilemma in the standard multi-armed bandit context [10, 8, 9]. In the context of “yes/no” rewards, a particularly useful, computable scheme is to maintain a Beta prior on each arm. This takes advantage of the conjugate nature of the Beta distribution for Bernoulli observations. The distribution $\beta(a, b)$ is updated to $\beta(a + 1, b)$ upon success and $\beta(a, b + 1)$ upon failure. For every pair (a, b) we can calculate the Gittins index $G(a, b)$. For simplicity we assume that when $a + b \geq 500$, the mean $\frac{a}{a+b}$ represents the correct probability of success for that arm. We choose the arm to play next by multiplying

Parameters: Price $Q \in [0.5, 2q^*]^K$, Matrix G of Gittins Indices.

Initialization: $n = 0$ (# buyers so far), Divide Q in 4 regions in increasing order of magnitude. Initialize state S for each of the K arms according to the region they lie in: from lower to higher: $(4,1), (3,2), (2,3), (1,4)$

For each k in Buyers do:

1. Price the item at Q_j which maximizes $Q_j \cdot G[S_j]$. Denote the chosen price by Q_{j^*} .
 2. If the buyer buys, set $S_j(a) = S_j(a) + 1$ else set $S_j(b) = S_j(b) + 1$
-

Table 1: A Gittins-Index Based Algorithm. The K parameter governs the discretization of the space (we use $K = 20$).

the Gittins index for each arm with its payoff if the arm is successful, $S_i = q_i G(a_i, b_i)$ and choosing the arm with highest S_i . This is equivalent to maintaining Gittins indices on arms with two payoffs, 0 and q_i [16].

The standard approach of initializing all the arms with the same prior is inappropriate in this case, because we know that the probability of a buyer buying at a higher price is lower. Thus we arrange the arms in increasing order of their weights and divide them in 4 region. We initialize arms in the region with lowest weight with a Beta $(4, 1)$ prior, the next lowest with a Beta $(3, 2)$ prior, next with $(2, 3)$ and the remaining with $(1, 4)$. As expected, this weighting of the priors significantly outperforms uniform priors on all the arms. Table 1 shows the final algorithm in detail.

UCB1.

Much work on digital goods auctions has focused on algorithms with good regret bounds. Two of these that are based on algorithms for multi-armed bandit problems have gained particular attention, namely the EXP3 algorithm [2, 3] and the UCB1 algorithm [1, 15]. Kleinberg discusses a “continuum armed” bandit algorithm called CAB1, which is

Parameters: Price $Q \in [0.5, 2q^*]^K$, Number of buyers: n_{ob} .

Initialization: $n = 0$ (# buyers so far)

For each k in first K buyers do:

1. Price the item at Q_k
2. $n_k = 1; n = n + 1$
3. If the buyer buys then $x_k = Q_k$ else $x_k = 0$

For the remaining buyers at each time instant t do:

1. Price the item at Q_j which maximizes $\frac{x_j}{n_j} + \sqrt{\frac{2 \ln n}{n_j}}$.
Denote the chosen price by Q_{j^*} .
 2. $n_{j^*} = n_{j^*} + 1; n = n + 1$
 3. If the buyer buys, set $x_{j^*} = x_{j^*} + Q_{j^*}$ and update total profit
-

Table 2: Algorithm UCB1, adapted to our setting. The K parameter governs the discretization of the space (we use $K = 20$).

a wrapper around algorithms like UCB1 or EXP3 for continuous spaces [14]. We perform extensive empirical tests on all these algorithms, adapted to our setting. UCB1 and EXP3 discretize the action space and treat each possible price as a unique possible action (or “arm” in bandit language). The EXP3 and UCB1 algorithms are specifically designed for adversarial and I.I.D. scenarios respectively. As expected, we find that EXP3 is outperformed (or equaled in performance) by UCB1 in all our I.I.D. scenarios, so we do not report results from EXP3. While one would expect CAB1 to perform well, since it is designed for continuous action spaces, it is geared more towards producing useful regret bounds, and does not take advantage of the structure of the search space, instead using doubling processes to efficiently scan a potentially large continuum. It is outperformed by UCB1. The specific form of the UCB1 algorithm we use is shown in Table 2.

4. EXPERIMENTAL RESULTS

We consider various different distributions that generate demand. We restrict ourselves to I.I.D. assumptions rather than considering adversarial scenarios.

Choice of distributions.

We consider three sets of valuation distributions that generate a wide range of optimal prices:

1. Uniform on $[0, B]$ where B is 4, 2.5, 1.5.
2. Exponential with rate (λ) parameters 1.75, 0.8, 0.5.
3. Log-normal with location (μ) and scale (σ) parameters (1, 1), (1, 0.75) and (1, 0.5).

Analysis of Results.

Each simulation consists of a stream of n buyers, arriving one after the other, each buyer has a valuation v that is sampled at random from the valuation distribution. The seller chooses a price q to offer, and if $v \geq q$ the buyer goes through with the purchase, otherwise she turns down the offer. In Figure 4 we report results averaged over 1000 simulations of the process, each consisting of 500 time steps.

In addition to comparing the algorithms, in cases where the linearity assumption of LLVD is violated (exponential and log-normal valuation distributions), we are interested in quantifying how much of the regret of the algorithm can be attributed to the linearity assumption itself, and how much may be due to not learning the best possible linear function. In order to study this, we also report the analytical profit that would be achieved by using the linear function of the form $1 - \gamma q$ to model the probability of buying, when γ is chosen so that the functional distance between the uniform distribution on $[0, 1/\gamma]$ and the true target valuation distribution is minimized. We evaluate functional distance between the two distributions as the sum of squared difference between their c.d.f (square of L2-Norm of the difference of the c.d.f). Let $F(x)$ and $G(x)$ be the two distributions

$$f_d = \text{L2-Norm} = \sqrt{\int_0^\infty (F(x) - G(x))^2 dx}$$

In our case where $F(x)$ is the uniform distribution in the interval $[0, B]$ where $B = 1/\gamma$.

$$D = f_d^2 = \int_0^B (F(x) - G(x))^2 dx + \int_B^\infty G^2(x) dx$$

Further details are in Appendix A.

Uniform valuation distributions (linear demand) As expected, LLVD always learns the correct distribution rapidly in these cases, significantly outperforming UCB1 and the Gittins-index based scheme.

Exponential valuation distributions In this case, $\Pr(\text{Buyer Buys}|q) = e^{-\lambda q}$, where λ is the rate parameter. LLVD performs either better than or as well as the Gittins-index based scheme in these cases, and significantly outperforms UCB1.

Log-normal valuation distributions For the log-normal, $\Pr(\text{Buyer Buys}|q) = 1 - \phi(\frac{\ln q - \mu}{\sigma})$, where μ and σ are the location and scale parameters for the log-normal distribution. While LLVD dominates UCB1, the Gittins-index based scheme is competitive, sometimes performing better and sometimes worse. LLVD may have trouble with these cases because the log-normal distribution is harder to approximate with a linear function, or because the learning process is thrown off. In some cases LLVD even outperforms the “best” linear function (indicating that the fit over the entire distribution is not necessarily the best measure when profit-seeking behavior is determined by only a portion of the distribution), providing evidence for the latter explanation.

A note about long-term learning.

It is worth noting that in the long-term, when the LLVD algorithm converges to a suboptimal price, it remains suboptimal, whereas bandit-based algorithms keep learning and slowly improving their performance over time. In some cases (like exponential distributions with $\lambda = 0.5, 0.8$) where LLVD and the Gittins index scheme perform similarly, the perfor-

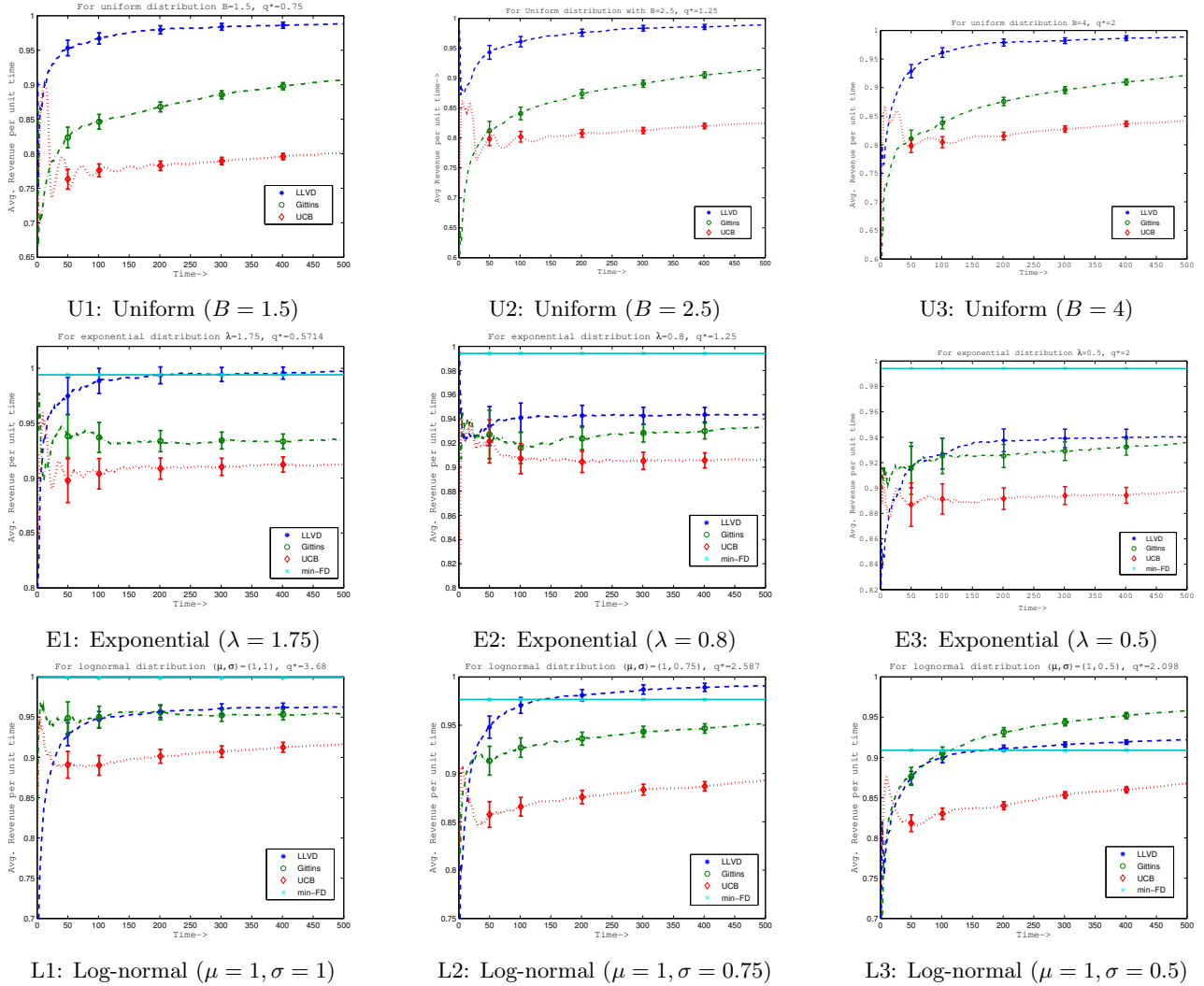


Figure 4: Main experimental results: Each graph shows the time-averaged profit received at any time, averaged over 1000 simulations and the 95% confidence interval. The top row shows uniform valuation distributions, corresponding to the model LLVD is based on. The second row shows exponential valuation distributions, and the bottom row log-normal ones. All values are represented as fraction of optimal profit.

mance of the Gittins index scheme continues to improve over time, eventually exceeding that of LLVD. Our primary interest is in maximizing revenue in the initial stages, because we assume that over time the distribution can be learned anyhow, perhaps in an “off-policy” manner.

5. DISCUSSION

As dynamic pricing becomes a reality with intelligent agents making rapid pricing decisions on the Internet, the field of algorithmic pricing has developed rapidly. While there has been continuing work on revenue management and inventory issues in operations research, the study of posted price mechanisms for digital goods auctions has mostly been confined to theoretical computer science, inspired by developments from computational learning theory. As a result, the focus has mostly been on deriving regret bounds rather than developing and analyzing algorithms that could prove

useful in practice. In the spirit of Vermorel and Mohri’s empirical analysis of algorithms for bandit problems [20], we believe that it is important to test algorithms in simulation, and ideally in real-world environments, or at least using real-world data. This paper starts exploring this path with simulation experiments.

We find that the UCB1 algorithm, which has some desirable theoretical properties for posted price auctions with unlimited supply, can be slow to learn in simple simulated environments; further, choosing the right number of arms can have a significant effect on performance (we experimented with several different numbers of arms to come up with a good number, reported in this paper). Theoretical extensions to spaces with a continuum of actions, like CAB1, fare no better. However, there are two promising directions: (1) an algorithm based on making a linearity assumption about the demand curve performs well, even when the true model

is not linear. Additionally, our experimental results and theoretical analysis of the linearity assumption indicate that it may be a very useful approximation, far beyond just for truly linear models. (2) Using simple but appropriate priors in a Gittins-index based scheme also shows promise. There is still scope to further improve performance by enabling better information sharing between arms. One possibility is to apply knowledge gradient techniques [18, 17] to the pricing problem, but current state-of-the-art KG techniques also do not account for correlation between arms. Existing extensions typically consider multivariate normal priors, though, which are not appropriate for monotonic functions like demand. This is a fruitful area for future work.

6. ACKNOWLEDGMENTS

We are grateful for research funding from an NSF CAREER award (0952918), and from a US-Israel BSF Grant (2008404). We thank David Sarne and Malik Magdon-Ismael for several helpful conversations.

7. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multiarmed bandit problem. *Machine Learning*, 47(2):235–256, 2002.

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proc. FOCS*, volume 36, pages 322–331. IEEE Computer Society Press, 1995.

[3] A. Blum, V. Kumar, A. Rudra, and F. Wu. Online learning in online auctions. *Theoretical Computer Science*, 324(2-3):137–146, 2004.

[4] T. Chakraborty, Z. Huang, and S. Khanna. Dynamic and non-uniform pricing strategies for revenue maximization. In *Proc. FOCS*, 2009.

[5] Y. Chen and R. Wang. Learning buyers’ valuation distribution in posted-price selling. *Economic Theory*, 14(2):417–428, 1999.

[6] V. Conitzer and N. Garera. Learning algorithms for online principal-agent problems (and selling goods online). In *Proceedings of the 23rd international conference on Machine learning*, pages 209–216. ACM, 2006.

[7] S. Das and M. Magdon-Ismael. Adapting to a market shock: Optimal sequential market-making. In *Advances in Neural Information Processing Systems (NIPS)*, pages 361–368, 2008.

[8] J. Gittins. Bandit processes and dynamic allocation indices. *Journal of the Royal Statistical Society. Series B (Methodological)*, 41(2):148–177, 1979.

[9] J. Gittins. *Multi-armed bandit allocation indices*. John Wiley & Sons Inc, 1989.

[10] J. Gittins and D. Jones. A dynamic allocation index for the discounted multiarmed bandit problem. *Biometrika*, 66(3):561–565, 1979.

[11] A. Goldberg and J. Hartline. Envy-free auctions for digital goods. In *Proc. ACM EC*, pages 29–35. ACM New York, NY, USA, 2003.

[12] A. Goldberg, J. Hartline, and A. Wright. Competitive auctions for multiple digital goods. In *Proc. ESA*, pages 416–427. Springer, 2001.

[13] J. Kephart, J. Hanson, and A. Greenwald. Dynamic pricing by software agents. *Computer Networks*, 32(6):731–752, 2000.

[14] R. Kleinberg. Nearly tight bounds for the continuum-armed bandit problem. *Advances in Neural Information Processing Systems*, 18, 2005.

[15] R. Kleinberg and T. Leighton. The value of knowing a demand curve: Bounds on regret for on-line posted-price auctions. In *Proc. FOCS*, 2003.

[16] M. Rothschild. A two-armed bandit theory of market pricing. *Journal of Economic Theory*, 9(2):185–202, 1974.

[17] I. Ryzhov, P. Frazier, and W. Powell. On the robustness of a one-period look-ahead policy in multi-armed bandit problems. *Procedia Computer Science*, 1(1):1629–1638, 2010.

[18] I. Ryzhov, W. Powell, and P. Frazier. The knowledge gradient algorithm for a general class of online learning problems. *Submitted for publication*, 2008.

[19] I. Segal. Optimal pricing mechanisms with unknown demand. *American Economic Review*, 93(3):509–529, 2003.

[20] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Proc. ECML*, pages 437–446. Springer, 2005.

APPENDIX

A. FUNCTIONAL DISTANCE

Let $F(x) = \frac{x}{B}$ represent the c.d.f of Uniform distribution over the interval $[0, B]$ and $G(x)$ be the c.d.f be the actual valuation distribution. L2-Norm for the difference between the two distributions is given by:

$$f_d = \sqrt{\int_0^\infty (F(x) - G(x))^2 dx}$$

For convenience we consider $D = f_d^2$, written as

$$D = f_d^2 = \int_0^\infty ((1 - G(x)) - (1 - F(x)))^2 dx$$

Let $F_1(x) = 1 - F(x)$ and $G_1(x) = 1 - G(x)$. Then

$$\begin{aligned} D &= \int_0^B F_1^2(x) dx - 2 \int_0^B G_1(x) F_1(x) dx + \int_0^\infty G_1^2(x) dx \\ &= \frac{B}{3} - 2 \int_0^B G_1(x) F_1(x) dx + \int_0^\infty G_1^2(x) dx \end{aligned}$$

differentiating w.r.t B and setting to 0 to calculate minima, we find

$$\frac{1}{3} - 2 \int_0^B \frac{qG_1(x)}{B^2} dx = 0$$

This equation can easily be solved numerically for $G(x)$ exponential and lognormal respectively, and it can be verified that $\frac{d^2 D}{dB^2} > 0$ for minima.

Ties Matter: Complexity of Voting Manipulation Revisited

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

Noam Hazon
Robotics Institute
Carnegie Mellon University,
USA
noamh@cs.cmu.edu

ABSTRACT

In their groundbreaking paper, Bartholdi, Tovey and Trick [1] argued that many well-known voting rules, such as Plurality, Borda, Copeland and Maximin are easy to manipulate. An important assumption made in that paper is that the manipulator’s goal is to ensure that his preferred candidate is among the candidates with the maximum score, or, equivalently, that ties are broken in favor of the manipulator’s preferred candidate. In this paper, we examine the role of this assumption in the easiness results of [1]. We observe that the algorithm presented in [1] extends to all rules that break ties according to a fixed ordering over the candidates. We then show that all scoring rules are easy to manipulate if the winner is selected from all tied candidates uniformly at random. This result extends to Maximin under an additional assumption on the manipulator’s utility function that is inspired by the original model of [1]. In contrast, we show that manipulation becomes hard when arbitrary polynomial-time tie-breaking rules are allowed, both for the rules considered in [1], and for a large class of scoring rules.

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, tie-breaking rules, complexity

1. INTRODUCTION

Computational social choice is an actively growing subarea of multiagent systems that provides theoretical foundations for preference aggregation and collective decision-making in multiagent domains. One of the most influential early contributions to this area is the paper by Bartholdi, Tovey, and Trick entitled “The computational difficulty of manipulating an election” [1]. In this paper, the authors suggested that computational complexity can serve as a barrier to dishonest behavior by the voters, and proposed classifying voting rules according to how difficult it is to manipulate

Cite as: Ties Matter: Complexity of Voting Manipulation Revisited, S. Obraztsova, E. Elkind, N. Hazon, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 71–78.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

them. In particular, they argued that such well-known voting rules as Plurality, Borda, Copeland and Maximin are easy to manipulate, yet a variant of the Copeland rule known as second-order Copeland is computationally resistant to manipulation. In a subsequent paper, Bartholdi and Orlin [2] showed that another well-known voting rule, namely, STV, is NP-hard to manipulate as well.

Since then, the computational complexity of manipulation under various voting rules, either by a single voter or by a coalition of voters, received considerable attention in the literature, both from the theoretical and from the experimental perspective (see, in particular, [20, 19] and the recent survey [9] for the former, and [17, 5] for the latter). While it has been argued that worst-case complexity does not provide adequate protection against malicious behavior (see, e.g. [15, 18, 10, 13]), determining whether a given voting rule is NP-hard to manipulate is still a natural first step in evaluating its resistance to manipulation in realistic scenarios.

An important property of the voting rules discussed in [1] is that they may produce multiple winners, i.e., they are, in fact, voting correspondences (see Section 2 for the formal definitions). It is not immediately clear what it means for manipulation to be successful in such a case. Bartholdi, Tovey and Trick take a rather liberal approach in their paper: they define a manipulation to be successful if, as a result, the manipulator’s preferred candidate is *one of the election winners*. This approach is equivalent to assuming that ties are broken in favor of the manipulator. Now, a careful examination of the algorithm in [1] shows that it works as long as ties are broken either adversarially to the manipulator or according to an arbitrary fixed lexicographic order over the candidates. However, in real-life settings, when an election ends in a tie, it is not uncommon to choose the winner using a tie-breaking rule that is non-lexicographic in nature. Indeed, perhaps the most common approach is to toss a coin, i.e., select the winner uniformly at random among all tied alternatives. A more sophisticated example is provided by the second-order Copeland rule studied in [1], which is effectively the Copeland rule combined with a rather involved tie-breaking method. Despite its apparent complexity, the second-order Copeland is the voting rule of choice for several organizations [1]. Thus, it is natural to ask under what conditions on the tie-breaking rule the voting correspondences considered in [1] remain easy to manipulate.

In this paper, we make two contributions towards answering this question. We first consider the randomized tie-breaking rule, which chooses the winner uniformly at random among all tied candidates. Now, to formalize the notion of a successful manipulation under this rule, we need additional information about the manipulator’s preferences: knowing the manipulator’s preference order is insufficient for determining whether he prefers a tie between his top candidate and his least favorite candidate to his second choice becom-

ing the unique winner. Thus, following [6], we endow the manipulator with utilities for all candidates, and seek a manipulation that maximizes his expected utility, where the expectation is taken over the random bits used to select the winner. We demonstrate that for all scoring rules such a manipulation can be found in polynomial time. This is also true for Maximin as long as the manipulator’s utility function has a special form that is inspired by the notion of manipulation employed in [1]: namely, the manipulator values one of the candidates at 1 and the rest of the candidates at 0.

Given these easiness results, it is natural to ask whether all (efficiently computable) tie-breaking rules produce easily manipulable rules when combined with the voting correspondences considered in [1]. Now, paper [1] shows that for Copeland this is not the case, by proving that the second-order Copeland rule is hard to manipulate. However, prior to our work, no such result was known for other rules considered in [1]. Our second contribution is in demonstrating that Maximin and Borda, as well as many families of scoring rules, become hard to manipulate if we allow arbitrary polynomial-time tie-breaking rules. This holds even if we require that the tie-breaking rule only depends on the set of the tied alternatives, rather than the voters’ preferences over them; we will refer to such tie-breaking rules as *simple*. Our proof also works for Copeland, thus strengthening the hardness result of [1] to simple tie-breaking rules. One can view these results as a continuation of the line of work suggested in [3, 7], namely, identifying minor tweaks to voting rules that make them hard to manipulate. Indeed, here we propose to “tweak” a voting rule by combining it with an appropriate tie-breaking rule; arguably, such a tweak affects the original rule less than the modifications proposed in [3] and [7] (i.e., combining a voting rule with a preround or taking a “hybrid” of the rule with itself or another rule). We remark, however, that our hardness result is not universal: Plurality and other scoring rules that correspond to scoring vectors with a bounded number of non-zero coordinates are easy to manipulate under any polynomial-time simple tie-breaking rule. However, if non-simple tie-breaking rules are allowed, Plurality can be shown to be hard to manipulate as well.

The rest of the paper is organized as follows. We cover the preliminaries and introduce the necessary notation in Section 2. Section 3 discusses the algorithm and the formal model of [1]. We describe the algorithms for scoring rules and Maximin under randomized tie-breaking in Section 4, and prove our hardness results in Section 5. Section 6 concludes.

2. PRELIMINARIES

An *election* is specified by a set of candidates C , $|C| = m$, and a set of voters $V = \{v_1, \dots, v_n\}$, where each voter v_i is associated with a linear order R_i over the candidates in C ; this order is called v_i ’s *preference order*. We denote the space of all linear orderings over C by $\mathcal{L}(C)$. For readability, we will sometimes denote R_i by \succ_i . When $a \succ_i b$ for some $a, b \in C$, we say that voter v_i prefers a to b . The vector $\mathcal{R} = (R_1, \dots, R_n)$, where each R_i is a linear order over C , is called a *preference profile*. A *voting rule* \mathcal{F} is a mapping that, given a preference profile \mathcal{R} over C outputs a candidate $c \in C$; we write $c = \mathcal{F}(\mathcal{R})$. Many classic voting rules, such as the ones defined below, are, in fact, *voting correspondences*, i.e., they map a preference profile \mathcal{R} to a non-empty subset S of C . Voting correspondences can be transformed into voting rules using tie-breaking rules. A *tie-breaking rule* for an election (C, V) 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 called *simple* if it does not depend on \mathcal{R} , i.e., the value of $T(\mathcal{R}, S)$ is uniquely determined by S . Such rules have the attractive property that if a manipulator can-

not change the set of tied candidates, he cannot affect the outcome of the election. Further, we say that 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 $a \in S \setminus \{c\}$.

A *composition* of a voting correspondence \mathcal{F} and a tie-breaking rule T is a voting rule $T \circ \mathcal{F}$ that, given a preference profile \mathcal{R} over C , outputs $T(\mathcal{R}, \mathcal{F}(\mathcal{R}))$. Clearly, $T \circ \mathcal{F}$ is a voting rule and $T \circ \mathcal{F}(\mathcal{R}) \in \mathcal{F}(\mathcal{R})$.

We will now describe the voting rules (correspondences) considered in this paper. All these rules assign scores to candidates; the winners are the candidates with the highest scores.

Scoring rules Any vector $\alpha = (\alpha_1, \dots, \alpha_m) \in \mathbb{R}^m$ with $\alpha_1 \geq \dots \geq \alpha_m$ defines a *scoring rule* \mathcal{F}_α . Under this rule, each voter grants α_i points to the candidate it ranks in the i -th position; the score of a candidate is the sum of the scores it receives from all voters. The vector α is called a *scoring vector*. A scoring rule is said to be *faithful* if $\alpha_1 > \dots > \alpha_m$. We are interested in scoring rules that are succinctly representable; therefore, throughout this paper we assume that the coordinates of α are nonnegative integers given in binary. We remark that scoring rules are defined for a fixed number of candidates. Therefore, we will often consider families of scoring rules, i.e., collections of the form $(\alpha^m)_{m=1}^\infty$, where $\alpha^m = (\alpha_1^m, \dots, \alpha_m^m)$. We require such families to be polynomial-time computable, i.e., we only consider families of voting rules $(\alpha^m)_{m=1}^\infty$ for which there exists a polynomial-time algorithm that given an $m \in \mathbb{N}$ outputs $\alpha_1^m, \dots, \alpha_m^m$. Two well-known examples of polynomial-time computable families of scoring rules are *Borda*, given by $\alpha^m = (m-1, \dots, 1, 0)$, and *k-approval*, given by $\alpha_i^m = 1$ if $i \leq k$, $\alpha_i^m = 0$ if $i > k$. 1-approval is also known as *Plurality*.

Copeland 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 . Given a rational value $\alpha \in [0, 1]$, under the 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., $\min_{d \in C \setminus \{c\}} |\{i \mid c \succ_i d\}|$.

Given a preference profile \mathcal{R} over a set of candidates C , for any preference order L over C we denote by (\mathcal{R}_{-i}, L) the preference profile obtained from \mathcal{R} by replacing R_i with L . We say that a voter v_i can successfully *manipulate* an election (C, V) with a preference profile (R_1, \dots, R_n) with respect to a voting rule \mathcal{F} if $\mathcal{F}(\mathcal{R}_{-i}, L) \succ_i \mathcal{F}(\mathcal{R})$. We will now define the computational problem that corresponds to this notion.

An instance of the \mathcal{F} -MANIPULATION problem is given by a set of candidates C , a set of voters V , a preference profile \mathcal{R} , and the manipulating voter v_i . It is a “yes”-instance if there exists a vote L such that $\mathcal{F}(\mathcal{R}_{-i}, L) \succ_i \mathcal{F}(\mathcal{R})$ and a “no”-instance otherwise.

3. THE MODEL AND THE ALGORITHM OF BARTHOLDI, TOVEY AND TRICK

Before we describe the algorithm presented in [1], we remark that the definition of successful manipulation given in [1] differs from our definition of \mathcal{F} -MANIPULATION (which is modeled after the standard social choice definition, see, e.g. [12, 16]), even if we assume that \mathcal{F} is a voting rule rather than a voting correspondence. Specifically, in [1] it is assumed that the manipulator has a preferred candidate p , and his goal is to make p elected; we will refer

to this problem as \mathcal{F} -MANIPULATION(p). However, a polynomial-time algorithm for \mathcal{F} -MANIPULATION(p) can be converted into a polynomial-time algorithm for \mathcal{F} -MANIPULATION: we can simply run \mathcal{F} -MANIPULATION(p) on all candidates ranked by the manipulator above the current winner, and pick the best among the candidates for which \mathcal{F} -MANIPULATION(p) outputs “yes”. Thus, if \mathcal{F} -MANIPULATION is hard, \mathcal{F} -MANIPULATION(p) is hard, too. Moreover, all of our hardness reductions directly show hardness of both variants of the problem.

The algorithm for \mathcal{F} -MANIPULATION(p) proposed in [1] assumes that the voting rule assigns scores to all candidates, and the winners are the candidates with the highest scores. Let v be the manipulator, and let p be her preferred candidate. The algorithm places p first, and then fills in the remaining positions in the vote from top to bottom, searching for a candidate that can be placed in the next available position in v ’s vote so that his score does not exceed that of p . This approach works as long as the rule is monotone and we can determine a candidate’s final score given his position in v ’s vote and the identities of the candidates that v ranks above him. It is not hard to show that Plurality and Borda (and, in fact, all scoring rules), as well as Copeland and Maximin have this property.

We can easily modify this algorithm for the setting where the ties are broken adversarially to the manipulator: in that case, when the manipulator fills a position i in his vote, $i > 1$, he needs to ensure that the score of the candidate in that position is strictly less than that of p . Generally, if ties are broken according to a lexicographic ordering \succ over the candidates, when placing a candidate c with $c \succ p$, the manipulator needs to make sure that c ’s score is less than that of p , and when placing a candidate c with $c \prec p$, he needs to make sure that c ’s score does not exceed that of p .

4. RANDOMIZED TIE-BREAKING RULES

In this section, we consider a very common approach to tie-breaking, namely, choosing the winner uniformly at random among all tied candidates. In this case, knowing the manipulator’s preference ordering is not sufficient to determine his optimal strategy. For example, suppose that voter v prefers a to b to c , and by voting strategically he can change the output of the voting correspondence from b to $\{a, c\}$. It is not immediately clear if this manipulation is beneficial. Indeed, if v strongly prefers a , but is essentially indifferent between b and c , then the answer is probably positive, but if v strongly dislikes c and slightly prefers a to b , the answer is likely to be negative (of course, this also depends on v ’s risk attitude).

Thus, to model this situation appropriately, we need to know the utilities that the manipulator assigns to all candidates. Under the natural assumption of risk neutrality, the manipulator’s utility for a set of candidates is equal to his expected utility when the candidate is drawn from this set uniformly at random, or, equivalently, to his *average* utility for a candidate in this set. Since we are interested in computational issues, it is reasonable to assume that all utilities are rational numbers; by scaling, we can assume that all utilities are positive integers given in binary.

Formally, given a set of candidates C , we assume that the manipulator is endowed with a utility function $u : C \rightarrow \mathbb{N}$. This function can be extended to sets of candidates by setting $u(S) = \frac{1}{|S|} \sum_{c \in S} u(c)$ for any $S \subseteq C$. Given a voting correspondence \mathcal{F} and an election (C, V) with a preference profile \mathcal{R} , we say that a vote L is *optimal* for a voter $v_i \in V$ with a utility function $u_i : C \rightarrow \mathbb{N}$ with respect to \mathcal{F} combined with the randomized tie-breaking rule if $u_i(\mathcal{F}(\mathcal{R}_{-i}, L)) \geq u_i(\mathcal{F}(\mathcal{R}_{-i}, L')$ for all $L' \in \mathcal{L}(C)$. We say that v_i has a *successful manipulation* if his optimal vote L satisfies $u_i(\mathcal{F}(\mathcal{R}_{-i}, L)) > u_i(\mathcal{F}(\mathcal{R}))$. In the rest of

this section, we will explore the complexity of finding an optimal vote with respect to scoring rules and Maximin.

4.1 Scoring rules

All scoring rules turn out to be easy to manipulate under randomized tie-breaking.

THEOREM 4.1. *For any election $E = (C, V)$ with $|C| = m$, any voter $v \in V$ with a utility function $u : C \rightarrow \mathbb{N}$, and any scoring vector $\alpha = (\alpha_1, \dots, \alpha_m)$, we can find in polynomial time an optimal vote for v with respect to the scoring rule \mathcal{F}_α combined with the randomized tie-breaking rule.*

PROOF. Fix a voter $v \in V$ with a utility function u , and let \mathcal{R}' denote the preference profile consisting of all other voters’ preferences. Let s_i denote the score of candidate c_i after all voters other than v have cast their vote. Let us renumber the candidates in order of increasing score, and, within each group with the same score, in order of decreasing utility. That is, under the new ordering we have $s_1 \leq \dots \leq s_m$ and if $s_i = s_j$ for some $i < j$ then $u(c_i) \geq u(c_j)$. We say that two candidates c_i, c_j with $s_i = s_j$ belong to the same *level*. Thus, all candidates are partitioned into $h \leq m$ levels H_1, \dots, H_h , so that if $c_i \in H_k$ and $c_j \in H_\ell$, $k < \ell$, then $s_i < s_j$.

Consider first the vote L_0 given by $c_1 \succ \dots \succ c_m$, and let T be the number of points obtained by the winner(s) in (\mathcal{R}', L_0) . We claim that for any $L \in \mathcal{L}(C)$, in the preference profile (\mathcal{R}', L) the winner(s) will get at least T points. Indeed, let c_i be the last candidate to get T points in (\mathcal{R}', L_0) , and suppose that there exists a vote L such that c_i gets less than T points in (\mathcal{R}', L) . By the pigeonhole principle, this means that L assigns at least α_i points to some c_j with $j > i$, and we have $s_j + \alpha_i \geq s_i + \alpha_i = T$, i.e., some other candidate gets at least T points, as claimed. We will say that a vote L is *conservative* if the winners’ score in (\mathcal{R}', L) is T .

We will now argue that if L maximizes v ’s utility, then either L is conservative or it can be chosen so that \mathcal{F}_α has a unique winner under (\mathcal{R}', L) . Indeed, suppose that this is not the case, i.e., any vote L that maximizes v ’s utility is such that the set $S = \mathcal{F}_\alpha(\mathcal{R}', L)$ is of size at least 2, and all candidates in S get $T' > T$ points. Let c_i be v ’s most preferred candidate in S ; we have $u(c_i) \geq u(S)$. Suppose that L grants α_j points to c_i . Since we have $c_i + \alpha_j > T$, it follows that $j < i$. Now, consider the vote obtained from L_0 by swapping c_i and c_j . Clearly, all candidates in $C \setminus \{c_i, c_j\}$ get at most T points, and c_i gets $T' > T$ points. Further, c_j gets $s_j + \alpha_i \leq s_j + \alpha_j \leq T$ points. Thus, in this case c_i is a unique winner and $u(c_i) \geq u(S)$, a contradiction.

Therefore, to find an optimal manipulation, it suffices to (i) check for each candidate $c \in C$ whether c can be made the unique winner with a score that exceeds T and (ii) find an optimal conservative vote. The optimal manipulation can then be selected from the ones found in (i) and (ii).

Step (i) is easy to implement. Indeed, a candidate c_i can be made the unique winner with a score that exceeds T if and only if $s_i + \alpha_1 > T$. To see this, observe that if $s_i + \alpha_1 > T$, we can swap c_1 and c_i in L_0 : c_i will get more than T points, and all other candidates will get at most T points. Conversely, if $s_i + \alpha_1 \leq T$, then the score of c_i is at most T no matter how v votes.

Thus, it remains to show how to implement (ii). Intuitively, our algorithm proceeds as follows. We start with the set of winners produced by L_0 ; we will later show that this set is minimal, in the sense that if it contains x candidates from some level, then for any vote the set of winners will contain at least x candidates from that level. Note also that due to the ordering of the candidates we select the best candidates from each level at this step. We then try

to increase the average utility of the winners' set. To this end, we order the remaining candidates by their utility, and try to add them to the set of winners one by one as long as this increases its average utility. We will now give a formal description of our algorithm and its proof of correctness.

Let $S_0 = \mathcal{F}_\alpha(\mathcal{R}', L_0)$. We initialize S and L by setting $S = S_0$, $L = L_0$. Let \succ^* be some ordering of the set C that ranks the candidates in S_0 first, followed by the candidates in $C \setminus S_0$ in the order of decreasing utility, breaking ties arbitrarily. We order the candidates from $C \setminus S_0$ according to \succ^* , and process the candidates in this ordering one by one. For each candidate c_i , we check if $u(c_i) > u(S)$; if this is not the case, we terminate, as all subsequent candidates have even lower utility. Otherwise, we check if we can swap c_i with another candidate that is currently not in S and receives $T - s_i$ points from L (so that c_i gets T points in the resulting vote). If this is the case, we update L by performing the swap and set $S = S \cup \{c_i\}$. We then proceed to the next candidate on the list.

We claim that the vote L obtained in the end of this process is optimal for the manipulator, among all conservative votes. We remark that at any point in time S is exactly the set of candidates that get T points in (\mathcal{R}', L) . Thus, we claim that any conservative vote \hat{L} satisfies $u(\mathcal{F}_\alpha(\mathcal{R}', \hat{L})) \leq u(S)$.

Assume that this is not the case. Among all optimal conservative votes, we will select one that is most "similar" to L in order to obtain a contradiction. Formally, let \mathcal{L}_0 be the set of all optimal conservative votes, and let \mathcal{L}_1 be the subset of \mathcal{L}_0 that consists of all votes L' that maximize the size of the set $\mathcal{F}_\alpha(\mathcal{R}', L') \cap S$. The ordering \succ^* induces a lexicographic ordering on the subsets of C . Let \hat{L} be the vote such that the set $\mathcal{F}_\alpha(\mathcal{R}', \hat{L})$ is minimal with respect to this ordering, over all votes in \mathcal{L}_1 . Set $\hat{S} = \mathcal{F}_\alpha(\mathcal{R}', \hat{L})$; by our assumption we have $u(\hat{S}) > u(S)$.

Observe first that our algorithm never removes a candidate from S : when we want to add c_i to S and search for an appropriate swap, we only consider candidates that have not been added to S yet. Also, at each step of our algorithm the utility of the set S strictly increases. These observations will be important for the analysis of our algorithm.

We will first show that $\hat{S} \setminus S$ is empty.

LEMMA 4.2. *We have $\hat{S} \setminus S = \emptyset$.*

PROOF. Suppose that the lemma is not true, and let c_i be a candidate in $\hat{S} \setminus S$. Suppose that c_i appears in the j -th position in our ordering of $C \setminus S_0$. If our algorithm terminated at or before the j -th step, we have $u(c_i) < u(S) < u(\hat{S})$, and hence $u(\hat{S} \setminus \{c_i\}) > u(\hat{S})$, a contradiction with the optimality of \hat{L} .

Thus, when our algorithm considered c_i , it could not find a suitable swap. Since $c_i \in \hat{S}$, it has to be the case that there exists an entry of the scoring vector that equals $T - s_i$; however, when our algorithm processed c_i it was unable to place c_i in a position that grants $T - s_i$ points. This could only happen if all candidates that were receiving $T - s_i$ points from L at that point were in S at that time; denote the set of all such candidates by B_i . Note that all candidates in B_i belong to the same level as c_i . Also, all candidates in $B_i \cap S_0$ have the same or higher utility than c_i , because initially we order the candidates at the same level by their utility, so that L_0 grants a higher score to the best candidates at each level. On the other hand, all candidates in $B_i \setminus S_0$ were added to S at some point, which means that they have been processed before c_i . Since at this stage of the algorithm we order the candidates by their utility, it means that they, too, have the same or higher utility than c_i .

Now, since \hat{L} grants $T - s_i$ points to c_i , it grants less than $T - s_i$ points to one of the candidates in B_i . Let c_k be any such candidate,

and consider the vote \hat{L}' obtained from \hat{L} by swapping c_i and c_k . Let $\hat{S}' = \mathcal{F}_\alpha(\mathcal{R}', \hat{L}')$; we have $\hat{S}' = (\hat{S} \setminus \{c_i\}) \cup \{c_k\}$. By the argument above, we have either $u(c_k) > u(c_i)$ or $u(c_k) = u(c_i)$. In the former case, we get $u(\hat{S}') > u(\hat{S})$. In the latter case, we get $u(\hat{S}') = u(\hat{S})$ and $|\hat{S}' \cap S| > |\hat{S} \cap S|$. In both cases, we obtain a contradiction with our choice of \hat{L} . \square

Thus, we have $\hat{S} \subseteq S$, and it remains to show that $S \subseteq \hat{S}$. We will first show that \hat{S} contains all candidates in S_0 .

LEMMA 4.3. *We have $S_0 \subseteq \hat{S}$.*

PROOF. Suppose that $|S_0 \cap H_k| = m_k$ for $k = 1, \dots, h$. We will first show that $|\hat{S} \cap H_k| \geq m_k$ for $k = 1, \dots, h$. Indeed, fix a $k \leq h$, and suppose that the first candidate in the k -th level is c_i . Then in (\mathcal{R}', L_0) the scores of the candidates in H_k are $s_i + \alpha_i, \dots, s_i + \alpha_j$ for some $j \geq i$. If $s_i + \alpha_i < T$, then $m_k = 0$ and our claim is trivially true for this value of k . Otherwise, by the pigeonhole principle, if it holds that in (\mathcal{R}', \hat{L}) less than m_k voters in H_k get T points, it has to be the case that at least one candidate in $H_{k+1} \cup \dots \cup H_h$ receives at least α_i points from \hat{L} . However, for any $c_\ell \in H_{k+1} \cup \dots \cup H_h$ we have $s_\ell > s_i$, so $s_\ell + \alpha_i > T$, a contradiction with our choice of \hat{L} .

Now, suppose that $S_0 \cap H_k \not\subseteq \hat{S} \cap H_k$ for some $k \leq h$, and consider a candidate $c_\ell \in (S_0 \cap H_k) \setminus (\hat{S} \cap H_k)$. Since we have argued that $|\hat{S} \cap H_k| \geq m_k$, it must be the case that there also exists a candidate $c_j \in (\hat{S} \cap H_k) \setminus (S_0 \cap H_k)$. It is easy to see that S_0 contains the m_k best candidates from H_k , so $u(c_\ell) \geq u(c_j)$. The rest of the proof is similar to that of Lemma 4.2: Consider the vote \hat{L}' obtained from \hat{L} by swapping c_ℓ and c_j and let $\hat{S}' = \mathcal{F}_\alpha(\mathcal{R}', \hat{L}')$. Since c_ℓ and c_j belong to the same level, we have $\hat{S}' = (\hat{S} \setminus \{c_\ell\}) \cup \{c_j\}$. Thus, either $u(\hat{S}') > u(\hat{S})$ or $u(\hat{S}') = u(\hat{S})$ and $|\hat{S}' \cap S| > |\hat{S} \cap S|$. In both cases we get a contradiction. Thus, we have $S_0 \cap H_k \subseteq \hat{S} \cap H_k$. Since this holds for every value of k , the proof is complete. \square

Given Lemma 4.2 and Lemma 4.3, it is easy to complete the proof. Suppose that \hat{S} is a strict subset of S . Observe first that for any subset S' of S there is a vote L' such that $\mathcal{F}_\alpha(\mathcal{R}', L') = S'$: we can simply ignore the candidates that are not members of S' when running our algorithm, as this only increases the number of "available" swaps at each step. Now, order the candidates in $C \setminus S_0$ according to \succ^* . Let c_i be the first candidate in this order that appears in S , but not in \hat{S} . If there is a candidate c_j that appears later in the sequence and is contained in both S and \hat{S} , consider the set $S' = \hat{S} \setminus \{c_j\} \cup \{c_i\}$. As argued above, there is a vote L' such that $\mathcal{F}_\alpha(\mathcal{R}', L') = S'$. Now, if $u(c_i) > u(c_j)$, this set has a higher average utility than \hat{S} . Thus, this is a contradiction with our choice of \hat{L} . On the other hand, if $u(c_j) = u(c_i)$, then we have $u(S') = u(\hat{S})$, $|S \cap S'| = |S \cap \hat{S}|$, and S' precedes \hat{S} in the lexicographic ordering induced by \succ^* , a contradiction with the choice of \hat{L} again. Therefore, none of the candidates in S that appear after c_i in the ordering belongs to \hat{S} . Now, when we added c_i to S , we did so because its utility was higher than the average utility of S at that point. However, by construction, the latter is exactly equal to $u(\hat{S})$. Thus, $u(\hat{S} \cup \{c_i\}) > u(\hat{S})$, a contradiction again. Therefore, the proof is complete. \square

4.2 Maximin

For Maximin with randomized tie-breaking, we have not been able to design an efficient algorithm for finding an optimal manipulation in the general utility model. However, we will now present a

polynomial-time algorithm for this problem assuming that the manipulator’s utility function has a special structure. Specifically, recall that in the model of [1] the manipulator’s goal is to make a specific candidate p a winner. This suggests that the manipulator’s utility can be modeled by setting $u(p) = 1$, $u(c) = 0$ for all $c \in C \setminus \{p\}$. We will now show that for such utilities there exists a poly-time algorithm for finding an optimal manipulation under Maximin combined with the randomized tie-breaking rule.

THEOREM 4.4. *If the manipulator’s utility function is given by $u(p) = 1$, $u(c) = 0$ for $c \in C \setminus \{p\}$, the problem of finding an optimal manipulation under Maximin combined with the randomized tie-breaking rule is in P.*

PROOF. Consider an election $E = (C, V)$ with the candidate set $C = \{c_1, \dots, c_m\}$ and the voter set $V = \{v_1, \dots, v_n\}$, and let v_n be the manipulating voter. In this proof, we denote by $s(c_i)$ the Maximin score of a candidate $c_i \in C$ in the election $E' = (C, V')$, where $V' = \{v_1, \dots, v_{n-1}\}$. Let $s = \max_{c_i \in C} s(c_i)$.

For any $c_i \in C$, the manipulator’s vote increases the score of c_i either by 0 or by 1. Thus, if $s(p) < s - 1$, the utility of the manipulator will be 0 irrespective of how he votes.

Now, suppose that $s(p) = s - 1$. The manipulator can increase the score of p by 1 by ranking p first. Thus, his goal is to ensure that after he votes (a) no other candidate gets $s + 1$ point and (b) the number of candidates in $C \setminus \{p\}$ with s points is as small as possible. Similarly, if $s(p) = s$, the manipulator can ensure that p gets $s + 1$ points by ranking him first, so his goal is to rank the remaining candidates so that in $C \setminus \{p\}$ the number of candidates with $s + 1$ points is as small as possible. We will now describe an algorithm that works for both of these cases.

We construct a directed graph G with the vertex set C that captures the relationship among the candidates. Namely, we have an edge from c_i to c_j if there are $s(c_j)$ voters in V' that rank c_j above c_i . Observe that, by construction, each vertex in G has at least one incoming edge. We say that c_i is a *parent* of c_j in G whenever there is an edge from c_i to c_j . We remark that if the manipulator ranks one of the parents of c_j above c_j in his vote, then c_j ’s score does not increase. We say that a vertex c_i of G is *purple* if $s(c_i) = s(p) + 1$, *red* if $s(c_i) = s(p)$ and $c_i \neq p$, and *green* otherwise; note that by construction p is green. Observe also that if $s(p) = s$, there are no purple vertices in the graph. We will say that a candidate c_j is *dominated* in an ordering L (with respect to G) if at least one of c_j ’s parents in G appears before c_j in L . Thus, our goal is to ensure that the set of dominated candidates includes all purple candidates and as many red candidates as possible.

Our algorithm is based on a recursive procedure \mathcal{A} , which takes as its input a graph H with a vertex set $U \subseteq C$ together with a coloring of U into green, red and purple; intuitively, U is the set of currently unranked candidates. It returns “no” if the candidates in U cannot be ranked so that all purple candidates in U are dominated by other candidates in U with respect to H . Otherwise, it returns an ordered list L of the candidates in U in which all purple candidates are dominated, and a set S consisting of all red candidates in U that remain undominated in L with respect to H .

To initialize the algorithm, we call $\mathcal{A}(G)$. The procedure $\mathcal{A}(H)$ is described below.

1. Set $L = \emptyset$.
2. If H contains p , set $L = [p]$, and remove p from H .
3. While H contains a candidate c that is green or has a parent that has already been ranked, set $L :: [c]$ (where $::$ denotes the list concatenation operation) and remove c from H .

4. If H is empty, return (L, \emptyset) .
5. If there is a purple candidate in H with no parents in H , return “no”.
6. If there is a red candidate c in H with no parents in H , let H' be the graph obtained from H by coloring c green. Compute $\mathcal{A}(H')$. If $\mathcal{A}(H')$ returns “no”, return “no”. Otherwise, if $\mathcal{A}(H')$ returns (L', S') , return $(L :: L', S' \cup \{c\})$.
7. At this point in the algorithm, each vertex of H has a parent. Hence, H contains a cycle. Let T be some such cycle. Collapse T , i.e., (a) replace T with a single vertex t , and (b) for each $y \notin T$, add an edge (t, y) if H contained an edge (x, y) for some $x \in T$ and add an edge (y, t) if H contained a vertex z with $(y, z) \in H$. Color t red if T contains at least one red vertex, and purple otherwise. Let H' be the resulting graph and call $\mathcal{A}(H')$. If $\mathcal{A}(H')$ returns “no”, return “no”. Now, suppose that $\mathcal{A}(H')$ returns (L', S') .

Suppose that $t \in S'$. At any point in the algorithm, we only put a vertex in S if it is red, so t must be red, and hence T contains a red vertex. Let c be some red vertex in T , and let \hat{L} be an ordering of the vertices in T that starts with c and follows the edges of T . Let L'' be the list obtained from L' by replacing t with \hat{L} (i.e., if $L' = L_1 :: [t] :: L_2$, then $L'' = L_1 :: \hat{L} :: L_2$). Return $(L :: L'', (S' \setminus \{t\}) \cup \{c\})$.

If $t \notin S'$, then by Lemma 4.5 (see below) t is dominated in H' . Let a be a parent of t that precedes it in L' . Then T contains a child of a . Let c be some such child, and let \hat{L} be an ordering of the vertices in T that starts with c and follows the edges of T . Let L'' be the list obtained from L' by replacing t with \hat{L} . Return $(L :: L'', S')$.

We will now argue that our algorithm outputs “no” if and only if no matter how v_n votes, some candidate in $C \setminus \{p\}$ gets $s(p) + 2$ points. Moreover, if $\mathcal{A}(G) = (L, S)$ and the set S contains r red candidates, then whenever v_n votes so that after his vote all other candidates have at most $s(p) + 1$ points, there are at least r red candidates with $s(p) + 1$ points.

We will split the proof into several lemmas.

LEMMA 4.5. *At any point in the execution of the algorithm, if $\mathcal{A}(H) = (L, S)$, then each candidate in $U \setminus S$ is dominated in H .*

PROOF. The proof is by induction on the recursion depth. Consider a candidate $x \in U \setminus S$. Clearly, if there are no recursive calls, \mathcal{A} ranks x at Step 3, and the claim is obviously true.

For the induction step, suppose that the claim is true if we have d nested recursive calls, and consider an execution that makes $d + 1$ nested calls. Again, consider a candidate $x \in U \setminus S$. As in the base case, if x has been ranked in Step 3 the claim is clearly true. If x was ranked in Step 6, it follows that $x \notin S'$, and the claim follows by the inductive assumption. Now, suppose that x was ranked in Step 7 when we collapsed some cycle T . If $x \notin T$, then $x \notin S'$ and the claim follows by the inductive assumption. In particular, if x was ranked after t before the expansion, there is some vertex y in T such that H contains the edge (y, x) , so after expansion x will be dominated by y .

Now, suppose that $x \in T$. If t was in S' , but x was not added to S , it means that x was not the first vertex of T to appear in the ranking, i.e., x was ranked after its predecessor in T . If t was not in S' , then by the inductive assumption t was ranked after its parent in H' , i.e., there is a $z \in H' \setminus \{t\}$ such that z is ranked before t in L' and there is an edge (z, t) in H' . By construction of t , this

means that there is a vertex $y \in T$ such that there is an edge (z, y) in H . Thus, when we expanded t into T , the first vertex of T to be ranked was placed after its parent, and all subsequent vertices of T were placed after their predecessors in T . Thus, all vertices in T and, in particular, x , are dominated. \square

We are now ready to prove that our algorithm correctly determines whether the manipulator can ensure that no candidate gets more than $s(p) + 1$ points.

LEMMA 4.6. *The algorithm outputs “no” if and only if for any vote L there is a purple candidate that is undominated.*

PROOF. Observe that the algorithm only outputs “no” if it finds a purple candidate with no parents. Let c be some such candidate. Now, in the original graph G each vertex has a parent. Further, if there was an edge from some x to c , and we collapsed a cycle T that contains x , but not c , there is still an edge from the resulting vertex t to c . Thus, the only way to obtain a purple vertex with no incoming edges is by collapsing a cycle T such that T contains purple vertices only, and no vertex of T has an incoming edge. By induction on the execution of the algorithm, it is easy to see that if we obtained a purple vertex with no incoming edges at some point, then in the original graph there was a group of purple vertices such that there was no edge from any red or green vertex to any of the vertices in the group. Now, in any ordering on C one of the candidates in this group would have to be ranked first. By construction, this candidate would be ranked before all its parents, so it is undominated.

Conversely, suppose that the algorithm does not answer “no”, and outputs a pair (L, S) instead. We have observed that S consists of red vertices only. Thus, by Lemma 4.5 each purple vertex is dominated. \square

It remains to show that the set S output by the algorithm contains as few candidates as possible.

LEMMA 4.7. *At any point in the execution of the algorithm, if $\mathcal{A}(H) = (L, S)$, then in any ordering of the candidates in U in which each purple vertex in U is dominated, at least $|S|$ red vertices in U are undominated.*

PROOF. The proof is by induction on the recursion depth. Suppose first that we make no recursive calls. Then our algorithm outputs $S = \emptyset$, and our claim is trivially true. Now, suppose that our claim is true if we make d nested calls. Consider an execution of \mathcal{A} which makes $d + 1$ nested calls, and suppose that when we call $\mathcal{A}(H')$ within this execution, it returns (L', S') .

Suppose first that we made the recursive call in Step 6 of the algorithm, and therefore set $S = S' \cup \{c\}$. Suppose for the sake of contradiction that there exists a ranking of the candidates in U such that at most $|S| - 1$ candidate is undominated. Since c has no parents in H , there are at most $|S| - 2$ other red candidates that are undominated. In other words, if we recolor c green, in the resulting instance (which is exactly the instance passed to \mathcal{A} during the recursive call), there are at most $|S| - 2$ undominated red candidates. Since $|S'| = |S| - 1$, this is a contradiction with the inductive assumption.

Now, suppose that we made the recursive call in Step 7 of the algorithm, and collapsed a cycle T into a vertex t . Again, assume for the sake of contradiction that there exists a ranking \bar{L} of the candidates in U such that at most $|S| - 1$ candidates are undominated. Let c be the first vertex of T to appear in \bar{L} . Consider the ranking of U' obtained by removing all vertices of $T \setminus \{c\}$ from \bar{L} and replacing c with t ; denote this ranking by \bar{L}' . We claim that in \bar{L}' at

most $|S| - 1$ vertices of H' are undominated. Indeed, any parent of c in H is a parent of t in H' , so t is undominated if and only if c was. On the other hand, if for some vertex x the only parent that preceded it in \bar{L} was a vertex $y \in T \setminus \{c\}$, then in H' there is an edge from t to x , i.e., x is preceded by its parent t in \bar{L}' . For all other vertices, if they were preceded by some parent z in \bar{L} , they are preceded by the same parent in \bar{L}' . Since $|S| = |S'|$, we have shown that U' can be ordered so that at most $|S'| - 1$ vertices are undominated, a contradiction with the inductive assumption. \square

Combining Lemma 4.6 and Lemma 4.7, we conclude that if our algorithm outputs (L, S) , then L is the optimal vote for v_n and if our algorithm outputs “no”, then the manipulator’s utility is 0 no matter how he votes. Also, it is not hard to see that the algorithm runs in polynomial time. Thus, the proof is complete. \square

5. HARDNESS RESULTS

We will now demonstrate that if we allow arbitrary polynomial-time tie-breaking rules, the algorithmic results presented in the previous sections no longer hold. In fact the problem of finding a beneficial manipulation becomes NP-hard. We will first present a specific simple tie-breaking rule T . We will then show that manipulating the composition of this rule with Borda, Copeland or Maximin is NP-hard.

Recall that an instance \mathcal{C} of 3-SAT is given by a set of s variables $X = \{x_1, \dots, x_s\}$ and a collection of t clauses $Cl = \{c_1, \dots, c_t\}$, where each clause $c_i \in Cl$ is a disjunction of three *literals* over X , i.e., variables or their negations; we denote the negation of x_i by \bar{x}_i . It is a “yes”-instance if there is a truth assignment for the variables in X such that all clauses in Cl are satisfied, and a “no”-instance otherwise. This problem is known to be hard even if we assume that all literals in each clause are distinct, so from now on we assume that this is the case. Now, given s variables x_1, \dots, x_s , there are exactly $\ell = \binom{2s}{3}$ 3-literal clauses that can be formed from these variables (this includes clauses of the form $x_1 \wedge \bar{x}_1 \wedge x_2$). Ordering the literals as $x_1 < \bar{x}_1 < \dots < x_s < \bar{x}_s$ induces a lexicographic ordering over all 3-literal clauses. Let ϕ_i denote the i -th clause in this ordering. Thus, we can encode an instance \mathcal{C} of 3-SAT with s variables as a binary string $\sigma(\mathcal{C})$ of length ℓ , where the i -th bit of $\sigma(\mathcal{C})$ is 1 if and only if ϕ_i appears in \mathcal{C} .

We are ready to describe T . Given a set $S \subseteq C$ of candidates, where $|C| = m$, T first checks if $m = \ell + 2s + 4$ for some $s > 0$ and $\ell = \binom{2s}{3}$. If this is not the case, it outputs the lexicographically first candidate in S and stops. Otherwise, it checks whether $c_m \in S$ and for every $i = 1, \dots, s$, the set S satisfies $|S \cap \{c_{\ell+2i-1}, c_{\ell+2i}\}| = 1$. If this is not the case, it outputs the lexicographically first candidate in S and stops. If the conditions above are satisfied, it constructs an instance $\mathcal{C} = (X, Cl)$ of 3-SAT by setting $X = \{x_1, \dots, x_s\}$, $Cl = \{\phi_i \mid 1 \leq i \leq \ell, c_i \in S\}$. Next, it constructs a truth assignment (ξ_1, \dots, ξ_s) for \mathcal{C} by setting $\xi_i = \top$ if $c_{\ell+2i-1} \in S$, $c_{\ell+2i} \notin S$ and $\xi_i = \perp$ if $c_{\ell+2i-1} \notin S$, $c_{\ell+2i} \in S$. Finally, if $\mathcal{C}(\xi_1, \dots, \xi_s) = \top$, it outputs c_m and otherwise it outputs the lexicographically first candidate in S . Clearly, T is simple and polynomial-time computable, and hence the problem $T \circ \mathcal{F}$ -MANIPULATION is in NP for any polynomial-time computable rule \mathcal{F} (and, in particular, for Borda, Maximin and Copeland). In the rest of this section, we will show that $T \circ \mathcal{F}$ -MANIPULATION is NP-hard for all these rules.

5.1 Borda and other scoring rules

We will first consider the Borda rule. We will then show that essentially the same proof works for a large class of scoring rules. To simplify notation, in the proof of Lemma 5.1 and Theorem 5.2

we will denote the Borda score of a candidate x in a preference profile \mathcal{R} by $s(\mathcal{R}, x)$.

LEMMA 5.1. *For any set of candidates $C = \{c_1, \dots, c_m\}$ with $m \geq 4$ and any vector $(\beta_1, \dots, \beta_{m-1})$ with $\beta_i \in \{0, 1, \dots, m\}$ for $i = 1, \dots, m-1$ and $\beta_1 > 0$, we can efficiently construct a preference profile $\mathcal{R} = (R_1, \dots, R_n)$ with $n = m(m-1)$ voters such that for some $K \geq m^2 + m + 1$ and some $u \leq m(m-1)$ the Borda scores of all candidates satisfy $s(\mathcal{R}, c_i) = K + \beta_i$ for $i = 1, \dots, m-1$ and $s(\mathcal{R}, c_m) = u$.*

The proof of Lemma 5.1 is similar to that of Theorem 3.1 in [5] and is omitted due to space constraints.

THEOREM 5.2. *$T \circ$ Borda-MANIPULATION is NP-hard.*

PROOF. Suppose that we are given an instance \mathcal{C} of 3-SAT with s variables. Note that this instance can be encoded by a binary vector $(\sigma_1, \dots, \sigma_\ell)$, where $\ell = \binom{2s}{3}$, as described in the construction of T : $\sigma_i = 1$ if and only if \mathcal{C} contains the i -th 3-variable clause with respect to the lexicographic order. We will now construct an instance of our problem with $m = \ell + 2s + 4$ candidates c_1, c_2, \dots, c_m . For readability, we will also denote the first ℓ candidates by u_1, \dots, u_ℓ , the next $2s$ candidates by $x_1, y_1, \dots, x_s, y_s$, and the last four candidates by d_1, d_2, w , and c .

Let $U = \{u_1, \dots, u_\ell\}$, let $Q = \{c_i \in U \mid \sigma_i = 1\}$, and let $q = |Q|$. For convenience, we renumber the candidates in U so that $Q = \{u_1, \dots, u_q\}$.

We will now use Lemma 5.1 to construct a preference profile $\mathcal{R} = (R_1, \dots, R_n)$ with the following scores:

- $s(\mathcal{R}, w) = K + m$, $s(\mathcal{R}, c) = K + 1$;
- $s(\mathcal{R}, u_i) = K + m - i$ for $i = 1, \dots, q$;
- $s(\mathcal{R}, u_i) = K$ for $i = q + 1, \dots, \ell$;
- $s(\mathcal{R}, x_i) = s(\mathcal{R}, y_i) = K + i + 1$ for $i = 1, \dots, s$;
- $s(\mathcal{R}, d_1) = K$, $s(\mathcal{R}, d_2) = u$,

where $K > m^2 + m + 1$ and $u \leq m(m-1)$.

Now, consider an election with the set of candidates C and a set of voters $V = \{v_1, \dots, v_{n+1}\}$, where for $i \leq n$ the preferences of the i -th voter are given by R_i , and the preferences of the last voter (who is also the manipulating voter) are given by

$$c \succ w \succ x_1 \succ y_1 \succ \dots \succ x_s \succ y_s \succ u_1 \succ \dots \succ u_\ell \succ d_1 \succ d_2.$$

Observe that if v_{n+1} votes truthfully, w wins. Thus, a manipulation is successful if and only if v_{n+1} manages to vote so that c gets elected.

Suppose first that we have started with a “yes”-instance of 3-SAT, and let $(\xi_1, \dots, \xi_s) \in \{\top, \perp\}^s$ be the corresponding truth assignment. For $i = 1, \dots, s$, set $z_i = x_i$ if $\xi_i = \top$ and $z_i = y_i$ if $\xi_i = \perp$. Suppose that v_{n+1} submits a vote L in which he ranks c, z_1, \dots, z_s in the top $s+1$ positions (in this order), u_q, \dots, u_1, w in the bottom $q+1$ positions (in this order), and all other candidates in the remaining positions in between.

It is not hard to see that in this case the candidates c, w, z_1, \dots, z_s and all candidates in Q get $K + m$ points, while all other candidates get less than $K + m$ points. Thus, the set of tied candidates S is $Q \cup \{c, w, z_1, \dots, z_s\}$. Therefore, given the set S , our tie-breaking rule will reconstruct \mathcal{C} , check whether z_1, \dots, z_s encode a satisfying truth assignment for \mathcal{C} (which is indeed the case), and output $c_m = c$. Thus, in this case L is a successful manipulation.

Conversely, suppose that v_{n+1} submits a vote L so that c gets elected. Since we have $s(\mathcal{R}, w) - s(\mathcal{R}, c) = m - 1$, it follows that L ranks c first and w last, and hence both of them get $K + m$

points. Similarly, we can show by induction on i that for all $i = 1, \dots, q$ it holds that v_{n+1} ranks u_i in the $(m - i)$ -th position; thus, each candidate in Q also gets $K + m$ points. Moreover, all other candidates in $U \setminus Q$ get less than $K + m$ points, i.e., the manipulator cannot change the formula encoded by the set of tied candidates. Let S be the set of all candidates with the top score. Since c wins the election, it has to be the case that the set $S \cap \{x_1, y_1, \dots, x_s, y_s\}$ encodes a satisfying truth assignment for \mathcal{C} , i.e., \mathcal{C} is satisfiable. Thus, the proof is complete. \square

Theorem 5.2 can be generalized to other families of scoring rules, including k -approval, where k is polynomially related to m (i.e., $m = \text{poly}(k)$). Note, however, that we cannot hope to prove an analogue of Theorem 5.2 for all scoring rules as long as we insist that the tie-breaking rule is simple: we have to require that the scoring vector has a superlogarithmic number of non-zero coordinates. Indeed, if the number of non-zero coordinates k satisfies $k = O(\log m)$, the manipulator can simply try all possible placements of the candidates into the top k positions in polynomial time. This strategy works for any simple polynomial-time tie-breaking rule, since the set of tied candidates only depends on the top k positions in the manipulator’s vote. On the other hand, if we drop the simplicity requirement, there are tie-breaking rules for which even Plurality is hard to manipulate.

THEOREM 5.3. *There exists a tie-breaking rule T' such that $T' \circ$ Plurality-MANIPULATION is NP-complete.*

We omit the formal proof of Theorem 5.3 due to space constraints; intuitively, we can consider a tie-breaking rule T' that interprets the set of winners as a boolean formula and views the manipulator’s vote as a truth assignment.

5.2 Copeland and Maximin

We will now show that $T \circ$ Copeland and $T \circ$ Maximin are hard to manipulate using essentially the same construction as in the proof of the Theorem 5.2.

THEOREM 5.4. *$T \circ$ Maximin-MANIPULATION is NP-hard.*

PROOF. Given a 3-SAT formula \mathcal{C} , we construct an election $E = (C, V)$ where C, U, Q and q are as in the proof of Theorem 5.2.

We can encode an election over a set of candidates C as a matrix $\{a(i, j)\}_{i, j \in C}$, where for all $i \neq j$ the entry $a(i, j)$ equals the number of voters that prefer i to j . By McGarvey’s theorem [14], for some $n = \text{poly}(m)$ we can efficiently construct a preference profile $\mathcal{R} = (R_1, \dots, R_n)$ corresponding to the following matrix:

- $a(u_i, u_{i+1}) = b + 1$ for $i = 1, \dots, q$;
- $a(u_i, u_{i+1}) = b - 1$ for $i = q + 1, \dots, \ell$;
- $a(x_i, y_i) = a(y_i, u_i) = b$ for $i = 1, \dots, s$;
- $a(c, w) = a(d_1, c) = b$, $a(w, d_1) = b + 1$;
- $a(c, d_2) = g + 1$;
- $a(x, y) = g + b - a(y, x)$ if $a(y, x)$ has been defined above;
- $a(x, y) = \frac{b+g}{2}$ for all other pairs $(x, y) \in C \times C$,

where $u_{\ell+1} := u_1$, $b < m$, $g > 2m$, and $b + g = n$. Now, consider an election with the set of candidates C and a set of voters $V = \{v_1, \dots, v_{n+1}\}$, where for $i \leq n$ the preferences of the i -th

voter are given by R_i , and the preferences of the last voter (who is also the manipulating voter) are given by

$$c \succ w \succ d_1 \succ d_2 \succ x_1 \succ y_1 \succ \dots \succ x_s \succ y_s \succ u_\ell \dots \succ u_1.$$

Observe that if v_{n+1} votes truthfully, then $a(w, d_1) = b + 2$, $a(w, x) > b + 2$ for all $x \in C \setminus \{d_1\}$, while the Maximin score of any other candidate is at most $b + 1$, so w is the election winner. Hence, a manipulation is successful if and only if v_{n+1} manages to vote so that c gets elected. It can be shown that this is possible if and only if we have started with a “yes”-instance of 3-SAT; we omit the proof due to lack of space. \square

THEOREM 5.5. $T \circ \text{Copeland}^\alpha$ -MANIPULATION is NP-hard for any $\alpha \in [0, 1]$.

PROOF. Given a 3-SAT formula \mathcal{C} , we construct an election $E = (C, V)$ where C, U, Q and q are the same as in the proof of the Theorem 5.2. We say that x safely wins a pairwise election against y (and y safely loses a pairwise election against x) if at least $\frac{n}{2} + 2$ voters prefer x to y . For any candidate $x \in C$, let $\text{SW}(x)$ and $\text{SL}(x)$ denote the number of pairwise elections that x safely wins and safely loses, respectively. By McGarvey’s theorem [14], we can construct a preference profile $\mathcal{R} = (R_1, \dots, R_n)$ with the following properties:

- $\text{SW}(u_i) = \frac{m+1}{2}, \text{SL}(u_i) = \frac{m-3}{2}$ for $i = 1, \dots, q$;
- $\text{SW}(u_i) = \frac{m-1}{2}, \text{SL}(u_i) = \frac{m-1}{2}$ for $i = q + 1, \dots, \ell$;
- $\text{SW}(x_i) = \text{SW}(y_i) = \frac{m-1}{2}$ for $i = 1, \dots, s$;
- $\text{SL}(x_i) = \text{SL}(y_i) = \frac{m-3}{2}$ for $i = 1, \dots, s$;
- $\text{SW}(c) = \frac{m-1}{2}, \text{SL}(c) = \frac{m-3}{2}$;
- $\text{SW}(w) = \frac{m+1}{2}, \text{SL}(w) = \frac{m-9}{2}$;
- $\text{SW}(d_1) = \lfloor \frac{\ell-q+s-1}{2} \rfloor, \text{SL}(d_1) = \lceil \frac{\ell+q+3s+5}{2} \rceil$;
- $\text{SW}(d_2) = \lceil \frac{\ell-q+s-1}{2} \rceil, \text{SL}(d_2) = \lfloor \frac{\ell+q+3s+5}{2} \rfloor$;
- there is a tie between c and w , w and d_1 , w and d_2 , and x_i and y_i for $i = 1, \dots, s$.

It is easy to check that for each candidate the total number of wins, losses and ties that involve him equals $m - 1$; in particular, $\ell - q + s - 1$ and $\ell + q + 3s + 5$ have the same parity, so $\text{SW}(d_1) + \text{SL}(d_1) = \text{SW}(d_2) + \text{SL}(d_2) = \ell + 2s + 2 = m - 2$. Moreover, the total number of wins equals the total number of losses. Thus, such a profile can indeed be constructed.

Now, consider an election with the set of candidates C and a set of voters $V = \{v_1, \dots, v_{n+1}\}$, where for $i \leq n$ the preferences of the i -th voter are given by R_i , and the preferences of the last voter (who is also the manipulating voter) are given by

$$c \succ w \succ d_1 \succ d_2 \succ x_1 \succ y_1 \succ \dots \succ x_s \succ y_s \succ u_\ell \dots \succ u_1.$$

If v_{n+1} votes truthfully, w wins. Hence, a manipulation is successful if and only if v_{n+1} manages to vote so that c gets elected. It can be shown that this is possible if and only if we started with a “yes”-instance of 3-SAT; we omit the proof due to lack of space. \square

6. CONCLUSIONS AND FUTURE WORK

We have explored the complexity of manipulating many common voting rules under randomized tie-breaking as well as under arbitrary polynomial-time tie-breaking procedures. Our results for randomized tie-breaking are far from complete, and a natural research direction is to extend them to other voting rules, such as Copeland or Bucklin, as well as to the Maximin rule with general utilities. Other interesting questions include identifying natural tie-breaking rules that make manipulation hard and extending our results to multi-winner elections.

7. ACKNOWLEDGMENTS

This research was supported by National Research Foundation (Singapore) under grant 2009-08, by NTU SUG (Edith Elkind), and by SINGA graduate fellowship (Svetlana Obraztsova).

8. REFERENCES

- [1] J. J. Bartholdi, III, C. A. Tovey, and M. Trick. The computational difficulty of manipulating an election. *Social Choice and Welfare*, 6:227–241, 1989.
- [2] J. J. Bartholdi, III and J. B. Orlin. Single transferable vote resists strategic voting. *Social Choice and Welfare*, 8(4):341–354, 1991.
- [3] V. Conitzer and T. Sandholm. Universal voting protocol tweaks to make manipulation hard. In *IJCAI’03*, pp. 781–788, 2003.
- [4] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate?. *J. ACM*, 54:1–33, 2007.
- [5] J. Davies, G. Katsirelos, N. Narodytska, and T. Walsh. An empirical study of Borda manipulation. In *COMSOC’10*, pp. 91–102, 2010.
- [6] Y. Desmedt, E. Elkind. Equilibria of plurality voting with abstentions. In *ACM EC’10*, pp. 347–356, 2010.
- [7] E. Elkind, H. Lipmaa, Hybrid voting protocols and hardness of manipulation. In *ISAAC’05*, pp. 206–215, 2005.
- [8] P. Faliszewski, E. Hemaspaandra, and H. Schnoor. Copeland voting: ties matter. In *AAMAS’08*, pp. 983–990, 2008.
- [9] P. Faliszewski, A. Procaccia. AI’s war on manipulation: are we winning?. In *AI Magazine*, 31(4):53–64, 2010.
- [10] E. Friedgut, G. Kalai, and N. Nisan. Elections can be manipulated often. In *FOCS’08*, pp. 243–249, 2008.
- [11] M. R. Garey, D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*, 1979.
- [12] A. F. Gibbard. Manipulation of voting schemes: a general result. *Econometrica*, 41:597–601, 1973.
- [13] M. Isaksson, G. Kindler, and E. Mossel. The geometry of manipulation—a quantitative proof of the Gibbard–Satterthwaite theorem. In *FOCS’10*, pp. 319–328, 2010.
- [14] D. C. McGarvey. A Theorem on the Construction of Voting Paradoxes. *Econometrica*, 21(4):608–610, 1953.
- [15] A. Procaccia, J. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *Journal of AI Research* 28:157–181, 2007.
- [16] M. A. Satterthwaite. Strategy-proofness and Arrow’s conditions: existence and correspondence theorems for voting procedures and social welfare functions. *Journal of Economic Theory*, 10:187–217, 1975.
- [17] T. Walsh. Where are the really hard manipulation problems? The phase transition in manipulating the Veto rule. In *IJCAI’09*, pp. 324–329, 2009.
- [18] L. Xia and V. Conitzer. A sufficient condition for voting rules to be frequently manipulable. In *EC’08*, pp. 99–108, 2008.
- [19] L. Xia, V. Conitzer, A. Procaccia. A scheduling approach to coalitional manipulation. In *EC’10*, pp. 275–284, 2010.
- [20] L. Xia, M. Zuckerman, A. Procaccia, V. Conitzer, and J. Rosenschein. Complexity of unweighted coalitional manipulation under some common voting rules. In *IJCAI’09*, pp. 348–352, 2009.

Designing Incentives for Boolean Games

Ulle Endriss* Sarit Kraus+ Jérôme Lang† Michael Wooldridge‡

*University of Amsterdam, The Netherlands (ulle.endriss@uva.nl)

+Bar Ilan University, Israel (sarit@cs.biu.ac.il)

†Université Paris-Dauphine, France (lang@irit.fr)

‡University of Liverpool, United Kingdom (mjw@liv.ac.uk)

ABSTRACT

Boolean games are a natural, compact, and expressive class of logic-based games, in which each player exercises unique control over some set of Boolean variables, and has some logical goal formula that it desires to be achieved. A player's strategy set is the set of all possible valuations that may be made to its variables. A player's goal formula may contain variables controlled by other agents, and in this case, it must reason strategically about how best to assign values to its variables. In the present paper, we consider the possibility of overlaying Boolean games with *taxation schemes*. A taxation scheme imposes a cost on every possible assignment an agent can make. By designing a taxation scheme appropriately, it is possible to perturb the preferences of the agents within a society, so that agents are rationally incentivised to choose some socially desirable equilibrium that would not otherwise be chosen, or incentivised to rule out some socially undesirable equilibria. After formally presenting the model, we explore some issues surrounding it (e.g., the complexity of finding a taxation scheme that implements some socially desirable outcome), and then discuss possible desirable properties of taxation schemes.

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, incentives, taxation

1. INTRODUCTION

The computational aspects of game-theoretic mechanism design have received a great deal of attention over the past decade [11]. Particular attention has been paid to the Vickrey-Clarke-Groves (VCG) mechanism, which can be used to incentivise rational agents to truthfully report their private preferences in settings such as combinatorial auctions [5, 10]. The key point of interest of the VCG mechanism is that, because it incentivises agents to report their

preferences truthfully, it allows us to compute outcomes that maximise social welfare, which would not in general be possible if agents could benefit from misrepresenting their preferences.

Ultimately, the VCG mechanism is a *taxation scheme*. Taxation schemes are used in human societies for several purposes. First, they are used to incentivise certain socially desirable behaviours, in much the same way that the VCG mechanism is used – for example, a government may tax car driving to encourage the use of environmentally friendly public transport. Second, they are used to raise revenue, typically with the intention that this revenue is then used to fund socially desirable projects (education, healthcare, etc). And finally, of course, they may be used for a combination of these purposes. Our aim in the present paper is to study the design of taxation schemes for incentivising behaviours in multi-agent systems. It is important to note that our focus in the present paper is *not* on the design of incentive compatible (truth-telling) mechanisms, and in this key respect, our work differs from the large body of work on computational and algorithmic mechanism design [11, 5, 10]. Of course, this is not to say that incentive compatibility is not important, or in any way to diminish the significance and value of the VCG mechanism; we are simply focussing on scenarios in which the preferences and actions of agents are known.

The setting for our study is the domain of *Boolean games* [6, 2, 4]. Boolean games are a natural, expressive, and compact class of games, based on propositional logic. Boolean games were introduced in [6], and their computational and logical properties have subsequently been studied by several researchers [2, 4]. In such a game, each agent i is assumed to have a goal, represented as a propositional formula γ_i over some set of variables Φ . In addition, each agent i is allocated some subset Φ_i of the variables Φ , with the idea being that the variables Φ_i are under the unique control of agent i . The choices, or strategies, available to i correspond to all the possible allocations of truth or falsity to the variables Φ_i . An agent will try to choose an allocation so as to satisfy its goal γ_i . Strategic concerns arise because whether i 's goal is in fact satisfied will depend on the choices made by others.

In the present paper, we introduce the idea of imposing *taxation schemes* on Boolean games, so that various possible choices are taxed in different ways. Taxation schemes are designed by an agent external to the system known as the *principal*. The ability to impose taxation schemes enables the principal to *perturb the preferences of the players in certain ways*: all other things being equal, an agent will prefer to make a choice that minimises taxes. As discussed above, the principal is assumed to be introducing a taxation scheme so as to incentivise agents to achieve a certain socially desirable outcome; or to incentivise agents to rule out certain socially undesirable outcomes. We represent the outcome that the principal desires to achieve via a propositional formula Υ : thus, the idea is

Cite as: Designing Incentives for Boolean Games, U. Endriss, S. Kraus, J. Lang, and M. Wooldridge, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonnenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 79–86. Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

that the principal will impose a taxation scheme so that agents are rationally incentivised to make individual choices so as to collectively satisfy Υ . However, a fundamentally important assumption in what follows is that taxes do not give us absolute control over an agent’s preferences. To assume that we were able to completely control an agent’s preferences by imposing taxes would be unrealistic: to pick a perhaps rather morbid and slightly tongue in cheek example, no matter how much you propose to tax me, I would still choose to achieve my goal of being alive rather than otherwise. If we *did* have complete control over agents’ preferences through taxation, then the problems we consider in this paper would indeed be rather trivial. In our setting specifically, it is assumed that no matter what the level of taxes, *an agent would still prefer to have its goal achieved than not*. This imposes a fundamental limit on the extent to which an agent’s preferences can be perturbed by taxation.

We begin in the following section by introducing the model of Boolean games that we use throughout the remainder of the paper. We then introduce taxation schemes, and the incentive design problem. After investigating some properties of the incentive design problem, we go on to consider socially equitable properties of taxation schemes (such as minimising the total tax burden, etc). We conclude with a discussion and future work.

2. BOOLEAN GAMES

In this section, we introduce the model of Boolean games that we work with throughout the remainder of this paper. This model slightly generalises previous models of Boolean games [6, 2, 4], in that it explicitly represents the costs of each action. In what follows, we let \mathbb{R}_{\geq} denote the set of real numbers greater than or equal to 0.

Propositional Logic: Throughout the paper, we make use of classical propositional logic, and for completeness, we thus begin by recalling the technical framework of this logic. Let $\mathbb{B} = \{\top, \perp\}$ be the set of Boolean truth values, with “ \top ” being truth and “ \perp ” being falsity. We will abuse notation a little by using \top and \perp to denote both the syntactic constants for truth and falsity respectively, as well as their semantic counterparts (i.e., the respective truth values). Let $\Phi = \{p, q, \dots\}$ 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 “ \perp ”. We assume a conventional semantic consequence relation “ \models ” for propositional logic. 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 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, i.e., is satisfied by every valuation. We denote the fact that formulae $\varphi, \psi \in \mathcal{L}$ are logically equivalent by $\varphi \Leftrightarrow \psi$; thus $\varphi \Leftrightarrow \psi$ means that $\models \varphi \leftrightarrow \psi$. Note that “ \Leftrightarrow ” is a meta-language relation symbol, which should not be confused with the object-language bi-conditional operator “ \leftrightarrow ”.

Agents, Goals, and Controlled Variables: The games we consider are populated by a set $Ag = \{1, \dots, n\}$ 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 $i \in Ag$. Each agent $i \in Ag$ *controls* a (possibly empty) subset Φ_i of the overall set of Boolean variables (cf. [14]). By “control”, we mean that i has the unique ability within the game to set the value (either \top or \perp) of each variable $p \in \Phi_i$. We will require that Φ_1, \dots, Φ_n forms

a partition of Φ , i.e., every variable is controlled by some agent and no variable is controlled by more than one agent ($\Phi_i \cap \Phi_j = \emptyset$ for $i \neq j$). Where $i \in Ag$, a *choice* for agent i is defined by 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 . The intuitive interpretation we give to \mathcal{V}_i is that it defines the *actions* or *strategies* available to agent i ; the *choices* available to the agent.

An *outcome*, $(v_1, \dots, v_n) \in \mathcal{V}_1 \times \dots \times \mathcal{V}_n$, is a collection of choices, one for each agent. Clearly, every outcome uniquely defines a valuation, and we will often think of outcomes as valuations, for example writing $(v_1, \dots, v_n) \models \varphi$ to mean that the valuation defined by the outcome (v_1, \dots, v_n) satisfies formula $\varphi \in \mathcal{L}$. Let $\varphi_{(v_1, \dots, v_n)}$ denote the formula that uniquely characterises the outcome (v_1, \dots, v_n) :

$$\varphi_{(v_1, \dots, v_n)} = \left(\bigwedge_{\substack{p \in \Phi \\ (v_1, \dots, v_n) \models p}} p \right) \wedge \left(\bigwedge_{\substack{q \in \Phi \\ (v_1, \dots, v_n) \not\models q}} \neg q \right)$$

Let $\text{succ}(v_1, \dots, v_n)$ denote the set of agents who have their goal achieved by outcome (v_1, \dots, v_n) , i.e.,:

$$\text{succ}(v_1, \dots, v_n) = \{i \in Ag \mid (v_1, \dots, v_n) \models \gamma_i\}.$$

Costs: Intuitively, the actions available to agents correspond to setting variables true or false. We assume that these actions have *costs*, defined by a *cost function* $c : \Phi \times \mathbb{B} \rightarrow \mathbb{R}_{\geq}$, so that $c(p, b)$ is the marginal cost of assigning variable $p \in \Phi$ the value $b \in \mathbb{B}$. We let c_0 denote the cost function that assigns zero cost to all assignments.

This notion of a cost function represents an obvious generalisation of previous presentations of Boolean games: costs were not considered in the original presentation of Boolean games [6, 2], and while costs were introduced in [4], it was assumed that only the action of setting a variable to \top would incur a cost. (In fact, as we shall see later, costs are, in a technical sense, not required in our framework; we can capture the key strategic issues at stake without them. However, it is natural from the point of view of modelling to have costs for actions, and to think about costs as being imposed from within the game, and taxes, (defined below), as being imposed from without.)

Boolean Games: Collecting these components together, a *Boolean game*, G , is a $(2n + 3)$ -tuple:

$$G = \langle Ag, \Phi, c, \gamma_1, \dots, \gamma_n, \Phi_1, \dots, \Phi_n \rangle,$$

where $Ag = \{1, \dots, n\}$ is a set of agents, $\Phi = \{p, q, \dots\}$ is a finite set of Boolean variables, $c : \Phi \times \mathbb{B} \rightarrow \mathbb{R}_{\geq}$ is a cost function, $\gamma_i \in \mathcal{L}$ is the goal of agent $i \in Ag$, and Φ_1, \dots, Φ_n is a partition of Φ over Ag , with the intended interpretation that Φ_i is the set of Boolean variables under the unique control of $i \in Ag$. We will say a game is *cost free* if it has cost function c_0 .

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 . Note that if an agent has multiple ways of getting its goal achieved, then it will prefer to choose one that minimises costs; and if an agent cannot get its goal achieved, then it simply chooses to minimise costs. These considerations are what give Boolean games their strategic character. For the moment, we will postpone the formal definition of the utility functions and preferences associated with our games.

3. DESIGNING INCENTIVES

We can now describe in more detail the overall problem that we consider in the remainder of the paper. Imagine a society populated by agents Ag , with each agent $i \in Ag$ having a goal $\gamma_i \in \mathcal{L}$ and actions corresponding to valuations to Φ_i . We assume an external *principal* has some goal $\Upsilon \in \mathcal{L}$ that it wants the society to achieve, and to this end, wants to incentivise the agents Ag to act collectively so as to bring about Υ . Incentives in our model are provided by *taxation schemes*.

Taxation Schemes: A taxation scheme defines additional (imposed) costs on actions, over and above those given by the marginal cost function c . While the cost function c is fixed and immutable for any given Boolean game, the principal is assumed to be at liberty to define a taxation scheme as they see fit. Agents will seek to minimise their overall costs, and so by assigning different levels of taxation to different actions, the principal can incentivise agents away from performing some actions and towards performing others; if the principal designs the taxation scheme correctly, then agents are incentivised to choose valuations (v_1, \dots, v_n) so as to satisfy Υ (i.e., so that $(v_1, \dots, v_n) \models \Upsilon$).

How exactly should we model taxation schemes? One very general approach would be to levy taxes on the basis of *outcomes*. We could model such taxes by a function $\tau : Ag \times \mathcal{V}_1 \times \dots \times \mathcal{V}_n \rightarrow \mathbb{R}_{\geq}$, with the intended interpretation that $\tau(i, v_1, \dots, v_n)$ is the amount of tax that would be imposed on agent i if the outcome (v_1, \dots, v_n) was selected. However, for the purposes of the present paper, we choose a simpler, *additive* model of taxes, the idea being that taxes are levied on individual actions, and the total tax imposed on an agent i is the sum of the taxes on individual choices (assignments of truth or falsity to a variable) made in the outcome v_i chosen by i .

Formally, we therefore model a taxation scheme as a function $\tau : \Phi \times \mathbb{B} \rightarrow \mathbb{R}_{\geq}$, where the intended interpretation is that $\tau(p, b)$ is the tax that would be imposed on the agent controlling p if the value b was assigned to the Boolean variable p . The total tax paid by an agent i in choosing a valuation $v_i \in \mathcal{V}_i$ will be $\sum_{p \in \Phi_i} \tau(p, v_i(p))$.

We let τ_0 denote the taxation scheme that applies no taxes to any choice, i.e., $\forall x \in \Phi$ and $b \in \mathbb{B}$, $\tau_0(x, b) = 0$. Let $\mathcal{T}(G)$ denote the set of taxation schemes over G . We make one technical assumption in what follows, relating to the space requirements for taxation schemes in $\mathcal{T}(G)$. Unless otherwise stated explicitly, we will assume that we are restricting our attention to taxation schemes whose values can be represented with a space requirement that is bounded by a polynomial in the size of the game. This seems a reasonable requirement: realistically, taxation schemes requiring space exponential in the size of the game at hand could not be manipulated. It is important to note that this requirement relates to the *space requirements for taxes*, and not to the *size of taxes themselves*: for a polynomial function $f : \mathbb{N} \rightarrow \mathbb{N}$, the value $2^{f(n)}$ can be represented using only a polynomial number of bits (i.e., $f(n)$ bits).

Utilities and Preferences: One important assumption we make is that while taxation schemes can influence the decision making of rational agents, they cannot, ultimately, change the goals of an agent. That is, if an agent has a chance to achieve its goal, it will take it, no matter what the taxation incentives are to do otherwise. To understand this point, and to see formally how incentives work, we need to formally define the utility functions for agents, and for this we require some further auxiliary definitions. First, with a slight abuse of notation, we extend cost and taxation functions to partial valuations as follows:

$$c_i(v_i) = \sum_{p \in \Phi_i} c(p, v_i(p))$$

$$\tau_i(v_i) = \sum_{p \in \Phi_i} \tau(p, v_i(p))$$

Next, let v_i^e denote the most expensive possible course of action for agent i :

$$v_i^e \in \arg \max_{v_i \in \mathcal{V}_i} (c_i(v_i) + \tau_i(v_i)).$$

Let μ_i denote the cost to i of its most expensive course of action:

$$\mu_i = c_i(v_i^e) + \tau_i(v_i^e).$$

Given these definitions, we define the *utility* to agent i of an outcome (v_1, \dots, v_n) , as follows:

$$u_i(v_1, \dots, v_n) = \begin{cases} 1 + \mu_i - (c_i(v_i) + \tau_i(v_i)) & \text{if } (v_1, \dots, v_n) \models \gamma_i \\ -(c_i(v_i) + \tau_i(v_i)) & \text{otherwise.} \end{cases}$$

Thus utility for agent i will range from $1 + \mu_i$ (the best outcome for i , where it gets its goal achieved by performing actions that have no tax or other cost) down to $-\mu_i$ (where i does not get its goal achieved but makes its most expensive choice). This definition has the following properties:

- an agent prefers all outcomes that satisfy its goal over all those that do not satisfy it;
- between two outcomes that satisfy its goal, an agent prefers the one that minimises total expense (= marginal costs + taxes); and
- between two valuations that *do not* satisfy its goal, an agent prefers to minimise total expense.

It is important to note that while utility functions provide a convenient numeric representation of preference relations, utility is not transferable in our settings.

Solution Concepts: Given this formal definition of utility, we can define solution concepts in the standard game-theoretic way [12]. In this paper, we focus on (pure) Nash equilibrium. (Of course, other solution concepts, such as dominant strategy equilibria, might also be considered, but for simplicity, in this paper we focus on Nash equilibria.) We say an outcome $(v_1, \dots, v_i, \dots, v_n)$ is a Nash equilibrium if for all agents $i \in Ag$, there is no $v_i' \in \mathcal{V}_i$ such that $u_i(v_1, \dots, v_i', \dots, v_n) > u_i(v_1, \dots, v_i, \dots, v_n)$. Let $NE(G, \tau)$ denote the set of all Nash equilibria of the game G with taxation scheme τ .

Before proceeding, let us consider some properties of Nash equilibrium outcomes. First, observe that an unsuccessful agent will choose a least cost course of action in any Nash equilibrium.

PROPOSITION 1. *Suppose $(v_1^*, \dots, v_i^*, \dots, v_n^*) \in NE(G, \tau)$ is such that $i \notin \text{succ}(v_1^*, \dots, v_i^*, \dots, v_n^*)$. Then*

$$v_i^* \in \arg \min_{v_i \in \mathcal{V}_i} c_i(v_i) + \tau_i(v_i)$$

PROOF. Agent i cannot make a choice v_i' that $(v_1^*, \dots, v_i', \dots, v_n^*) \models \gamma_i$, otherwise $u_i(v_1^*, \dots, v_i', \dots, v_n^*) > u_i(v_1^*, \dots, v_i^*, \dots, v_n^*)$, in which case $(v_1^*, \dots, v_i^*, \dots, v_n^*) \notin NE(G, \tau)$. So, the only way i could profitably deviate would be by making an alternative choice v_i' that reduced costs compared to v_i^* . But by definition, v_i^* minimises i 's costs. \square

The following is an obvious decision problem:

NASH OUTCOME VERIFICATION:

Instance: Boolean game G , taxation scheme τ , and outcome (v_1, \dots, v_n) .

Question: Is $(v_1, \dots, v_n) \in NE(G, \tau)$?

PROPOSITION 2. NASH OUTCOME VERIFICATION is co-NP-complete, even for two player games with $\tau = \tau_0$ and where c assigns no costs.

PROOF. Membership is immediate. For hardness, we reduce SAT to the complement problem. Given an instance φ of SAT over variables x_1, \dots, x_k , define a game G with $Ag = \{1, 2\}$, $\Phi = \{x_1, \dots, x_k, z\}$, (where z does not occur in φ), $\Phi_1 = \{x_1, \dots, x_k\}$, $\Phi_2 = \{z\}$, $\gamma_1 = \varphi$, $\gamma_2 = z$, let $v_1(y) = \perp$ for all $y \in \Phi_1$, and let $v_2(z) = \top$. We claim $(v_1, v_2) \notin NE(G, \tau_0)$ iff φ is satisfiable. (\rightarrow) Suppose $(v_1, v_2) \notin NE(G, \tau_0)$. Then either agent 1 or agent 2 can benefit by deviating. Clearly agent 2 cannot benefit, since it gets its goal achieved through v_2 at no cost, which is optimal for 2. So 1 must be able to benefit by deviating. Since it incurs no cost through v_1 , the only way agent 1 could benefit would be by achieving its goal, which would imply φ was satisfiable. (\leftarrow) Suppose φ is satisfiable. Then player 1 could benefit by choosing a valuation $v'_1 \neq v_1$ satisfying φ . Hence $(v_1, v_2) \notin NE(G, \tau_0)$. \square

Next, note that while being able to model costs in games explicitly is attractive from a modelling perspective, it is, in a sense, unnecessary from a purely technical point of view: we can always design a taxation scheme that simulates the costs and thus gives rise to the same set of Nash equilibria.

PROPOSITION 3. Let G be a game with cost function c and let τ be a taxation scheme for G . Then there exists a taxation scheme τ' such that $NE(G, \tau) = NE(G', \tau')$ for the game G' we obtain by replacing c with c_0 in G .

PROOF. Let $\tau'(p, b) = \tau(p, b) + c(p, b)$ for all $p \in \Phi$ and all $b \in \mathbb{B}$. Then the utility functions for (G', τ') are identical to those for (G, τ) , and thus the Nash equilibria must coincide as well. \square

Moreover, we can show that, for the analysis of Nash equilibria, it suffices to consider taxation schemes that only impose taxes on making a variable *true* (rather than *false*). Call a taxation scheme τ *positive* if $\tau(p, \perp) = 0$ for all $p \in \Phi$. Now consider two zero cost games G and G^1 . We call G^1 a *variant* of G if G^1 is the same as G , except that for some $p \in \Phi$ all occurrences of p in the agents' goals γ_i have been replaced by $\neg p$ (but Υ has not been changed).

PROPOSITION 4. Let G be a zero cost game and let τ be a taxation scheme for that game. Then there exists a variant G^1 of G and a positive taxation scheme τ' such that $NE(G, \tau) = NE(G^1, \tau')$.

PROOF. We have to define G^1 and τ' with respect to each $p \in \Phi$. For all variables p we will have $\tau'(p, \perp) = 0$ (as τ' should be positive). So we have to define the values $\tau'(p, \top)$ and we have to specify whether p should occur in the goal formulas in G^1 as in G , or whether p should get flipped (i.e., whether it should get rewritten as $\neg p$).

1. If $\tau(p, \top) = \tau(p, \perp)$, then we set $\tau'(p, \top) = 0$ and we leave p untouched in the game.

¹This restriction to games with zero cost is not required, but it does simplify exposition; and we have just seen that for the analysis of Nash equilibria it suffices to consider zero cost games.

2. If $\tau(p, \top) > \tau(p, \perp)$, then we set $\tau'(p, \top) = \tau(p, \top) - \tau(p, \perp)$ and we again leave p untouched in the game.
3. If $\tau(p, \top) < \tau(p, \perp)$, then we set $\tau'(p, \top) = \tau(p, \perp) - \tau(p, \top)$ and we flip p in the game.

The crucial feature of this construction is that the difference in tax between making p *true* or *false* remains $|\tau(p, \top) - \tau(p, \perp)|$ in the new game, and the new taxation scheme still “pushes in the same direction” as before. Therefore, the utility functions for (G', τ') are identical to those for (G, τ) , and thus the Nash equilibria must coincide as well. \square

Incentive Design: We now come to the main problems that we consider in the remainder of the paper. Suppose we have an agent, which we will call the principal, who is external to a game G . The principal is at liberty to impose taxation schemes on the game G . It will not do this for no reason, however: it does it because it wants to provide incentives for the agents in G to choose certain collective outcomes. Specifically, the principal wants to incentivise the players in G to choose rationally a collective outcome that satisfies an *objective*, which is represented as a propositional formula Υ over the variables Φ of G . We refer to this general problem – trying to find a taxation scheme that will incentivise players to choose rationally a collective outcome that satisfies a propositional formula Υ – as the *implementation problem*. It inherits concepts from the theory of Nash implementation in mechanism design [7], although our use of Boolean games, taxation schemes, and propositional formulae to represent objectives is quite different.

3.1 Weak Implementation

Let $\mathcal{WI}(G, \Upsilon)$ denote the set of taxation schemes over G that satisfy a propositional objective Υ in at least one Nash equilibrium outcome:

$$\mathcal{WI}(G, \Upsilon) = \{\tau \in \mathcal{T}(G) \mid \exists (v_1, \dots, v_n) \in NE(G, \tau) \text{ s.t. } (v_1, \dots, v_n) \models \Upsilon\}.$$

Given this definition, we can state the first basic decision problem that we consider in the remainder of the paper:

WEAK IMPLEMENTATION:

Instance: Boolean game G and objective $\Upsilon \in \mathcal{L}$.

Question: Is it the case that $\mathcal{WI}(G, \Upsilon) \neq \emptyset$?

If the answer to the WEAK IMPLEMENTATION problem (G, Υ) is “yes”, then we say that Υ *can be weakly implemented in Nash equilibrium* (or simply: Υ can be weakly implemented in G). Let us see an example.

EXAMPLE 1. Define a game G as follows: $Ag = \{1, 2\}$, $\Phi = \{p_1, p_2\}$, $\Phi_i = \{p_i\}$, $\gamma_1 = p_1$, $\gamma_2 = \neg p_1 \wedge \neg p_2$, $c(p_1, b) = 0$ for all $b \in \mathbb{B}$, while $c(p_2, \top) = 1$ and $c(p_2, \perp) = 0$. Define an objective $\Upsilon = p_1 \wedge p_2$. Now, without any taxes (i.e., with taxation scheme τ_0), there is a single Nash equilibrium, (v_1^*, v_2^*) , which satisfies $p_1 \wedge \neg p_2$. Agent 1 gets its goal achieved, while agent 2 does not; and moreover $(v_1^*, v_2^*) \not\models \Upsilon$. However, if we adjust τ so that $\tau(p_2, \perp) = 10$, then we find a Nash equilibrium outcome (v'_1, v'_2) such that $(v'_1, v'_2) \models p_1 \wedge p_2$, i.e., $(v'_1, v'_2) \models \Upsilon$. Here, agent 2 is not able to get its goal achieved, but it can, nevertheless, be incentivised by taxation to make a choice that ensures the achievement of the objective Υ .

So, what objectives Υ can be weakly implemented? At first sight, it might appear that the satisfiability of Υ is a sufficient condition for

implementability. Consider the following naive approach for constructing taxation schemes with the aim of implementing satisfiable objectives Υ :

(*) Find a valuation v such that $v \models \Upsilon$ (such a valuation will exist since Υ is satisfiable). Then define a taxation scheme τ such that $\tau(p, b) = 0$ if $b = v(p)$ and $\tau(p, b) = k$ otherwise, where k is a suitably large number.

Thus, the idea is simply to make all choices other than selecting an outcome that satisfies Υ too expensive to be rational. In fact, this approach does not work, because of an important subtlety of the definition of utility. In designing a taxation scheme, the principal can perturb an agent's choices between different valuations, but it *cannot* perturb them in such a way that an agent would prefer an outcome that does not satisfy its goal over an outcome that does. We have:

PROPOSITION 5. *There exist instances of the WEAK IMPLEMENTATION problem with satisfiable objectives Υ that cannot be weakly implemented.*

PROOF. Consider the following example. Define a game G as follows: $Ag = \{1\}$, $\Phi = \Phi_1 = \{p\}$, $\gamma_1 = p$, $c(p, b) = 0$ for all $b \in \mathbb{B}$. Let $\Upsilon = \neg p$. Suppose there is a taxation scheme τ such that $\exists v_1 \in NE(G, \tau)$ and $v_1 \models \Upsilon$. Clearly, $v_1(p) = \perp$, so $v_1 \not\models \gamma_1$ and thus $u_1(v) = -(c_1(v_1) + \tau(v_1)) = 0$. But consider the valuation $v'_1(p) = \top$, which since $v'_1 \models \gamma_1$ would yield $u_1(v'_1) = 1 + \mu_1 - (c_1(v'_1) + \tau_1(v'_1)) = 1$. Thus $u_1(v'_1) > u_1(v_1)$; contradiction. \square

What about tautologous objectives, i.e., objectives Υ such that $\Upsilon \Leftrightarrow \top$? Again, we might be tempted to assume that tautologies are trivially implementable. This is not in fact the case, however, as it may be that $NE(G, \tau) = \emptyset$ for all taxation schemes τ :

PROPOSITION 6. *There exist instances of the WEAK IMPLEMENTATION problem with tautologous objectives Υ that cannot be implemented.*

PROOF. Define a game G with $Ag = \{1, 2\}$, $\Phi = \{p, q\}$, $\Phi_1 = \{p\}$, $\Phi_2 = \{q\}$, $\gamma_1 = (p \leftrightarrow q)$, $\gamma_2 = \neg(p \leftrightarrow q)$, and c assigns zero cost to all actions. Clearly $NE(G, \tau) = \emptyset$. For example, in the outcome (v_1, v_2) in which $v_1(p) = \top$ and $v_2(q) = \top$, agent 1 would prefer to change its valuation to $v'_2(q) = \top$. There is, in fact, no taxation scheme τ such that $NE(G, \tau) \neq \emptyset$. \square

Tautologous objectives might appear to be of little interest, but we argue that this is not the case. Suppose we have a game G such that $NE(G, \tau_0) = \emptyset$. Then, in its unmodified condition, this game is *unstable*: it has no equilibria. Thus, we will refer to the problem of implementing \top (= checking for the existence of a taxation scheme that would ensure at least one Nash equilibrium outcome), as the STABILISATION problem. The following example illustrates STABILISATION.

EXAMPLE 2. *Let $Ag = \{1, 2, 3\}$, with $\varphi = \{p, q, r\}$, $\Phi_1 = \{p\}$, $\Phi_2 = \{q\}$, $\Phi_3 = \{r\}$, $\gamma_1 = \top$, $\gamma_2 = (q \wedge \neg p) \vee (q \leftrightarrow r)$, $\gamma_3 = (r \wedge \neg p) \vee \neg(q \leftrightarrow r)$, $c(p, \top) = 0$, $c(p, \perp) = 1$, and all other costs are 0. For any outcome in which $p = \perp$, agent 1 would prefer to set $p = \top$, so no such outcome can be stable. So, consider outcomes (v_1, v_2, v_3) in which $p = \top$. Here if $(v_1, v_2, v_3) \models q \leftrightarrow r$ then agent 3 would prefer to deviate, while if $(v_1, v_2, v_3) \not\models q \leftrightarrow r$ then agent 2 would prefer to deviate. Now, consider a taxation scheme with $\tau(p, \top) = 10$ and $\tau(p, \perp) = 0$ and all other taxes are 0. With this scheme, the outcome in which all variables are set to \perp is a Nash equilibrium. Hence this taxation scheme stabilises the system.*

Returning to the weak implementation problem, we can derive a *sufficient* condition for weak implementation, as follows.

PROPOSITION 7. *For all games G and objectives Υ , if the formula Υ' is satisfiable:*

$$\Upsilon' = \Upsilon \wedge \bigwedge_{i \in Ag} \gamma_i$$

then $WI(G, \Upsilon) \neq \emptyset$.

PROOF. Assume $\Upsilon' = \Upsilon \wedge \bigwedge_{i \in Ag} \gamma_i$ is satisfiable. Let v be a valuation such that $v \models \Upsilon'$. The basic idea is to use the approach (*), described above, to build a taxation scheme ensuring that the valuation v is a rational choice. For all $i \in Ag$, $x \in \Phi_i$ and $b \in \mathbb{B}$, define:

$$\tau(x, b) = \begin{cases} 0 & \text{if } b = v(x) \\ 1 + c_i(v_i^e) & \text{otherwise.} \end{cases}$$

(Recall that v_i^e is the choice for i that has the highest marginal cost.) Let (v_1^*, \dots, v_n^*) be the outcome corresponding to the valuation v . Obviously, $(v_1^*, \dots, v_n^*) \models \Upsilon$. We claim that $(v_1^*, \dots, v_n^*) \in NE(G, \tau)$. For suppose that (v_1^*, \dots, v_n^*) is not a Nash equilibrium. Then some agent i can benefit by deviating. Since by construction $(v_1^*, \dots, v_n^*) \models \gamma_i$, then i can only benefit from a choice that would decrease its overall costs. But the construction of τ ensures that any other choice would *increase* taxes more than any benefit gained by decreasing marginal costs. So, i cannot benefit by changing its choice, and so (v_1^*, \dots, v_n^*) is a Nash equilibrium. \square

We know from [2] that the problem of checking for the existence of pure strategy Nash equilibria in cost-free Boolean games is Σ_2^P -complete. It turns out that the IMPLEMENTATION problem is no harder:

PROPOSITION 8. *The STABILISATION problem is Σ_2^P -complete, even if taxes are 0-bounded. As a consequence, the WEAK IMPLEMENTATION problem is also Σ_2^P -complete.*

PROOF. Membership requires evaluating the following condition:

$$\exists \tau \in \mathcal{T}(G), \exists (v_1, \dots, v_n) \in \mathcal{V}_1 \times \dots \times \mathcal{V}_n, \\ \underbrace{(v_1, \dots, v_n) \in NE(G, \tau)}_{(**)}$$

Notice that the condition (**) is a co-NP predicate, and that the existential quantifiers can be computed in NP (recall that we assume taxation schemes in $\mathcal{T}(G)$ require space at most polynomial in the size of G , and hence guessed in non-deterministic polynomial time). Thus the problem is in Σ_2^P . For hardness, we can trivially reduce the problem of checking for the existence of pure strategy Nash equilibria in cost-free Boolean games, which was proved Σ_2^P -complete in [2, Proposition 5]. Given a cost free game as in [2], we construct an instance of one of our games directly, setting all costs to 0; we then ask whether the system can be stabilised with a tax bound of 0. Clearly, the answer is “yes” iff the given Boolean game instance has a pure strategy Nash equilibrium. \square

3.2 (Strong) Implementation

The fact that $WI(G, \Upsilon) \neq \emptyset$ is good news of a kind – it tells us that we can impose a taxation scheme such that *at least one* rational (NE) outcome of the game satisfies Υ . However, it could be that there are many taxation schemes, and only one of them satisfies Υ . This motivates us to consider the *strong implementation* (or simply *implementation*) problem. Strong implementation corresponds closely to the notion of Nash implementation in the mechanism design literature [7]. Let $\mathcal{SI}(G, \Upsilon)$ denote the set of taxation schemes τ over G such that:

1. G, τ has at least one Nash equilibrium outcome;
2. all Nash equilibrium outcomes of G, τ satisfy Υ .

Formally:

$$\begin{aligned} \mathcal{SI}(G, \Upsilon) = & \\ \{ \tau \in \mathcal{T}(G) \mid & \\ NE(G, \tau) \neq \emptyset \ \& \\ \forall (v_1, \dots, v_n) \in NE(G, \tau) : (v_1, \dots, v_n) \models \Upsilon \}. & \end{aligned}$$

This gives us the following decision problem:

IMPLEMENTATION:

Instance: Boolean game G and objective $\Upsilon \in \mathcal{L}$.

Question: Is it the case that $\mathcal{SI}(G, \Upsilon) \neq \emptyset$?

It turns out that strong implementation is no harder than weak implementation:

PROPOSITION 9. IMPLEMENTATION is Σ_2^p -complete.

PROOF. Observe that the problem involves evaluating the following condition: is it the case that $\exists \tau \in \mathcal{T}(G) : NE(G, \tau) \neq \emptyset$ and $\forall (v_1, \dots, v_n) \in NE(G, \tau)$ we have $(v_1, \dots, v_n) \models \Upsilon$? Expanding out and re-arranging, it can be seen that this is equivalent to asking whether $\exists \tau \in \mathcal{T}(G), \exists (v_1, \dots, v_n) \in \mathcal{V}_1 \times \dots \times \mathcal{V}_n, \forall (v'_1, \dots, v'_n) \in \mathcal{V}_1 \times \dots \times \mathcal{V}_n$, we have $(v_1, \dots, v_n) \in NE(G, \tau)$ and if $(v'_1, \dots, v'_n) \in NE(G, \tau)$ we have $(v'_1, \dots, v'_n) \models \Upsilon$. Clearly this is a Σ_2^p predicate. For hardness, we can reduce the STABILISATION problem as in Proposition 8. \square

How are $\mathcal{WI}(G, \Upsilon)$ and $\mathcal{SI}(G, \Upsilon)$ related? It turns out that weak and strong implementation are indeed different:

PROPOSITION 10.

1. For all games G and objectives Υ we have:

$$\mathcal{SI}(G, \Upsilon) \subseteq \mathcal{WI}(G, \Upsilon).$$

2. There exist games G and objectives Υ s.t.:

$$\mathcal{WI}(G, \Upsilon) \not\subseteq \mathcal{SI}(G, \Upsilon).$$

PROOF. Item (1) is immediate: if a taxation scheme strongly implements Υ in G then it weakly implements it. For item (2), we give an example of an game and objective such that the objective can be weakly, but not strongly implemented. Let $Ag = \{1, 2\}$, with $\Phi = \{p, q\}$, $\Phi_1 = \{p\}$, $\Phi_2 = \{q\}$, $\gamma_1 = \gamma_2 = (p \leftrightarrow q)$, with cost function c_0 . Finally, let $\Upsilon = p \wedge q$. Now, the taxation function τ_0 with zero taxes will weakly implement Υ : there will be two Nash equilibria, one satisfying $p \wedge q$ and the other satisfying $\neg(p \vee q)$. However, $\Upsilon = p \wedge q$ cannot be strongly implemented, because the outcome satisfying $\neg(p \vee q)$ will be a Nash equilibrium for all taxation schemes τ . To see this, observe that for it not to be a Nash equilibrium, one agent would benefit by deviating; but by definition of the utility functions, such a deviation would involve an agent moving from positive to negative utility. \square

Thus, to show that an objective Υ cannot be strongly implemented, it suffices to show that it cannot be weakly implemented.

One interesting question relates to the size of taxes required to for implementation. In some cases, it turns out that we only require very small amounts of tax:

PROPOSITION 11. Let $\varepsilon \ll 1$ be any arbitrarily small positive number. If G is cost-free (i.e., with cost function c_0) then Υ is implementable (respectively, weakly implementable) in G iff Υ is implementable by a taxation scheme bounded by ε .

PROOF. We first claim that for any arbitrarily small ε and any cost-free Boolean game G there is a cost-free Boolean game such that the taxation scheme is bounded by ε and the sets of Nash equilibria of the two games coincide. The idea is to define a new taxation scheme τ' by using ε to systematically scale down taxation values from τ . The transformation from (G, τ) to (G, τ') preserves the relative order on utilities, that is, $u_i^{G, \tau}(v_i) \geq u_i^{G, \tau'}(v_i)$ if and only if $u_i^{G, \tau'}(v_i) \geq u_i^{G, \tau'}(v'_i)$. Therefore, $v = (v_1, \dots, v_n)$ is a Nash equilibrium of (G, τ) if and only if it is a Nash equilibrium of (G, τ') . As a consequence, Υ is implementable (respectively, weakly implementable) in G by taxation scheme τ if and only if it is implementable (resp. weakly implementable) in G by τ' . \square

4. TAXATION AND SOCIAL WELFARE

In attempting to design a taxation scheme τ for a Boolean game G , the primary aim of a principal is to design the scheme so that agents are rationally motivated to choose an outcome satisfying the objective Υ . However, if it is possible to incentivise agents to satisfy Υ , then there will, in general, be multiple possible taxation schemes that incentivise the agents in this way, and not all of these taxation schemes will be equally desirable from the point of view of society. In this section, therefore, we consider different societal criteria that might be considered by a principal when choosing a taxation scheme; our discussion here is inspired by the literature on axioms for cooperative decision-making [9].

Utilitarian Social Welfare: The first idea we consider is the very well-known concept of *maximising utilitarian social welfare*. Formally, the social welfare of an outcome (v_1, \dots, v_n) is denoted $sw(v_1, \dots, v_n)$:

$$usw(v_1, \dots, v_n) = \sum_{i \in Ag} u_i(v_1, \dots, v_n).$$

An outcome $(v_1^{usw}, \dots, v_n^{usw})$ that maximises utilitarian social welfare is thus one satisfying:

$$(v_1^{usw}, \dots, v_n^{usw}) \in \arg \max_{(v_1, \dots, v_n)} usw(v_1, \dots, v_n).$$

Of course, simply finding an outcome that maximises social welfare in itself is not much use if agents are rationally motivated to choose another outcome, which does not maximise social welfare. We therefore say a taxation scheme τ *weakly implements* utilitarian social welfare maximisation in a game G if

$$\begin{aligned} \exists (v'_1, \dots, v'_n) \in NE(G, \tau) \text{ s.t.} \\ (v'_1, \dots, v'_n) \in \arg \max_{(v_1, \dots, v_n)} usw(v_1, \dots, v_n). \end{aligned}$$

We can define strong implementation in the expected way. With a slight abuse of notation, let $\mathcal{WI}(G, usw)$ denote the set of taxation schemes that weakly implement utilitarian social welfare maximisation, and let $\mathcal{SI}(G, usw)$ denote the set of taxation schemes that strongly implement utilitarian social welfare maximisation. Can we always implement utilitarian social welfare maximisation? No:

PROPOSITION 12. There are games G in which utilitarian social welfare maximisation cannot be weakly implemented (and hence cannot be strongly implemented).

PROOF. Define a game G with $Ag = \{1, 2, 3\}$, $\Phi = \{p_1, p_2, p_3\}$, $\Phi_i = \{p_i\}$, and $\gamma_1 = p_1 \vee (p_2 \wedge p_3)$, $\gamma_2 = \neg p_2$, $\gamma_3 = \neg p_3$, $c(p_1, \top) = 20$, $c(p_1, \perp) = 1$, $c(p_2, \top) = 2$, $c(p_2, \perp) = 1$, $c(p_3, \top) = 2$, $c(p_3, \perp) = 1$. Now, with taxation scheme τ_0 , there is a unique Nash equilibrium (v_1^*, v_2^*, v_3^*) in which agent 1 sets $p_1 = \top$, agent 2 sets $p_2 = \perp$, agent 3 sets $p_3 = \perp$. We have $u_1(v_1^*, v_2^*, v_3^*) = 1 + 20 - 20 = 1$, $u_2(v_1^*, v_2^*, v_3^*) = 1 + 2 - 1 = 2$,

and $u_3(v_1^*, v_2^*, v_3^*) = 1 + 2 - 1 = 2$, and so $usw(v_1^*, v_2^*, v_3^*) = 1 + 2 + 2 = 5$. Now consider the outcome (v_1, v_2, v_3) that satisfies $(\neg p_1) \wedge p_2 \wedge p_3$. Observe that $(v_1, v_2, v_3) \models \gamma_1$, while $(v_1, v_2, v_3) \not\models (\gamma_2 \vee \gamma_3)$. We have $u_1(v) = 1 + 20 - 1 = 20$, $u_2(v) = -2$, and $u_3(v) = -2$. Thus $usw(v_1, v_2, v_3) = 20 + (-2) + (-2) = 16$. Clearly, outcome (v_1, v_2, v_3) maximises social welfare, but no taxation scheme can weakly implement this outcome: agents 2 and 3 will always prefer to get their goal achieved. \square

This example also illustrates that maximising utilitarian social welfare is not the same as maximising the number of agents that get their goal achieved.

Notice that because we represent objectives Υ as logical formula, and these logical formula can completely characterise outcomes, we can directly model the problem of implementing utilitarian social welfare maximisation as a WEAK IMPLEMENTATION problem. The following is immediate:

PROPOSITION 13. *It is possible to weakly (respectively, strongly) implement utilitarian social welfare maximisation in game G iff $\mathcal{WI}(G, \Upsilon_{usw}) \neq \emptyset$ (respectively, $\mathcal{SI}(G, \Upsilon_{usw}) \neq \emptyset$), where:*

$$\Upsilon_{usw} = \bigvee_{(v_1^*, \dots, v_n^*) \in \arg \max_{(v_1, \dots, v_n) \in \mathcal{V}_1 \times \dots \times \mathcal{V}_n} usw(v_1, \dots, v_n)} \varphi(v_1^*, \dots, v_n^*).$$

From the point of view of a principal, of course, the main concern is to implement an objective Υ ; maximising utilitarian social welfare is a secondary concern. A very natural aim of the principal will therefore be to design a taxation scheme that implements an objective while at the same time maximises the worst case utilitarian social welfare of all possible Nash equilibrium outcomes. This yields the following decision problem,

USW IMPLEMENTATION:

Instance: Boolean game G , objective Υ , social welfare measure $w \in \mathbb{R}$.

Question: Does there exist a taxation scheme $\tau \in \mathcal{SI}(G, \Upsilon)$ such that $\min\{usw(v, \dots, v_n) \mid (v_1, \dots, v_n) \in NE(G, \tau)\} \geq w$?

PROPOSITION 14. *The USW IMPLEMENTATION problem is Σ_2^P -complete.*

PROOF. We reduce WEAK IMPLEMENTATION. Where G, Υ be an instance of WEAK IMPLEMENTATION, we create an instance G, Υ, w of USW IMPLEMENTATION with a value for w that is guaranteed to be below the worst case utilitarian social welfare of any outcome. \square

The corresponding function problem is to compute a taxation scheme maximising the worst case utilitarian social welfare of a Nash equilibrium outcome satisfying Υ . We will denote such a taxation scheme by $\tau_{usw}(G, \Upsilon)$:

$$\tau_{usw}(G, \Upsilon) \in \arg \max_{\tau \in \mathcal{SI}(G, \Upsilon)} \min\{usw(v, \dots, v_n) \mid (v_1, \dots, v_n) \in NE(G, \tau)\}.$$

Notice that in the case $\Upsilon \equiv \top$, finding $\tau_{usw}(G, \Upsilon)$ reduces to simply implementing utilitarian social welfare maximisation.

Egalitarian Social Welfare: A standard criticism of utilitarian social welfare is that it does not consider how utility is distributed amongst members of a society; it may allocate all utility to one agent, leaving all others with no utility. Egalitarian social welfare provides an alternative metric: it looks at how well off the least

well off member of society is. The function $esw(\dots)$ gives the egalitarian social welfare of an outcome:

$$esw(v_1, \dots, v_n) = \min\{u_i(v_1, \dots, v_n) \mid i \in Ag\}.$$

The taxation scheme implementing Υ while maximising egalitarian social welfare in G is denoted $\tau_{esw}(G, \Upsilon)$:

$$\tau_{esw}(G, \Upsilon) \in \arg \max_{\tau \in \mathcal{SI}(G, \Upsilon)} \min\{esw(v, \dots, v_n) \mid (v_1, \dots, v_n) \in NE(G, \tau)\}.$$

Minimising the Total Tax Burden: An alternative to measuring social welfare is to consider developing a taxation scheme that implements objective Υ while imposing the lowest possible tax burden on society. Broadly, we can think of this approach as minimising the degree of intervention of the principal in the operation of society. The function $tb(\dots)$ gives the total tax burden of an outcome:

$$tb(v_1, \dots, v_n) = \sum_{i \in Ag} \tau(v_i).$$

We define τ_{tb} in the obvious way:

$$\tau_{tb} \in \arg \min_{\tau \in \mathcal{SI}(G, \Upsilon)} \max\{tb(v_1, \dots, v_n) \mid (v_1, \dots, v_n) \in NE(G, \tau)\}$$

It is easy to construct examples showing that minimising the total tax burden may result in socially undesirable outcomes (but nevertheless, it seems such “least intervention” approaches are relatively popular in human societies).

5. TAXATION AND EQUITY

It is, of course, well-known that an outcome which maximises (for example) utilitarian social welfare may in fact be extremely undesirable from the point of view of the majority of agents in a system. For example, the social welfare maximising outcome might allocate all the utility in the system to one agent, leaving all others with none. This motivates us to consider a range of possible other notions of equity with respect to taxation schemes, inspired to some extent by the economics literature on taxation [3].

Minimising the Difference in Taxes: One very obvious (although arguably naive) notion of taxation equity is to simply try to ensure that agents are taxed at broadly the same level, i.e., to minimise the maximum difference in taxes levied on different agents. Let $md(v_1, \dots, v_n)$ give the maximum difference in taxes between any two agents in outcome (v_1, \dots, v_n) :

$$md(v_1, \dots, v_n) = \max\{abs(\tau_i(v_i) - \tau_j(v_j)) \mid \{i, j\} \subseteq Ag\}$$

where $abs(x)$ denotes the absolute value of x . Let τ_{md} denote a taxation scheme that minimises this value over all possible Nash equilibria of taxation schemes that implement Υ in G :

$$\tau_{md} \in \arg \min_{\tau \in \mathcal{SI}(G, \Upsilon)} \max\{md(v_1, \dots, v_n) \mid (v_1, \dots, v_n) \in NE(G, \tau)\}$$

Horizontal Equity: Simply aiming to apply the same level of taxes across an entire society may appear to be equitable, but on closer examination, it has some definite drawbacks. In particular, it does not distinguish between agents that have their goals achieved and those that do not. In the literature on taxation, the term *horizontal equity* is used to describe the idea that those in the same circumstances should be taxed at the same level [3]. One could formalise this notion in several different ways for our model, but we will fo-

cus on the following idea: in any outcome, we have two “classes” of agents: those that get their goal achieved and those that do not. Thus, when looking at the differences in taxes paid, we only compare the taxes of agents that get their goal achieved against other agents that get their goal achieved, and we compare only compare agents that do not get their goal achieved against agents that do not get their goal achieved. The function $he(\dots)$ denote the maximum difference in tax paid between agents in the same equivalence class:

$$he(v_1, \dots, v_n) = \max(\{abs(\tau_i(v_i) - \tau_j(v_j)) \mid \{i, j\} \subseteq Ag \ \& \ (v_1, \dots, v_n) \models \gamma_i \wedge \gamma_j\} \cup \{abs(\tau_i(v_i) - \tau_j(v_j)) \mid \{i, j\} \subseteq Ag \ \& \ (v_1, \dots, v_n) \models \neg(\gamma_i \vee \gamma_j)\})$$

Then τ_{he} will denote an outcome that maximises horizontal equity (i.e., minimises the difference in taxes paid by agents in the same circumstances).

$$\tau_{md} \in \arg \min_{\tau \in \mathcal{S}\mathcal{I}(G, \Upsilon)} \max\{he(v_1, \dots, v_n) \mid (v_1, \dots, v_n) \in NE(G, \tau)\}$$

6. CONCLUSIONS & FUTURE WORK

Taxation schemes, in the form of the VCG mechanism and variations thereof, have received an enormous amount of attention in the computer science literature over the past two decades [11]. Much of the current interest stems from the possibility, provided by VCG, of having incentive compatible mechanisms, i.e., mechanisms that incentivise agents to truthfully report their preferences, thereby allowing the computation of outcomes that maximise social welfare. There are, however, fundamental limits to what can be achieved with incentive compatible mechanisms, and it therefore seems worth considering the design of taxation schemes to incentivise behaviours in non incentive compatible settings. After all, taxation schemes in the real world are rarely incentive compatible. In the present paper, we have studied the use of taxation schemes to incentivise behaviours in Boolean games: a natural, expressive, and compact class of logic-based games. We showed how a principal could perturb the preferences of agents in a Boolean game by imposing a taxation scheme, and in so doing, how it could, in certain circumstances, incentivise agents to choose outcomes to satisfy some social objective Υ , represented as a Boolean formula. However, we saw that while an agent’s preferences can be perturbed, they are not completely malleable: no matter what the taxation scheme, an agent would always prefer to get its goal achieved than otherwise. This means there are limits on the extent to which preferences can be perturbed by taxation, and hence limits on what objectives Υ can be achieved. We studied a number of questions around the question of implementing objectives Υ via taxation schemes, and also discussed some issues surrounding equitable taxation.

Our work relates to a number of other topics in the multi-agent systems community and beyond. Some consideration has been given to how a principal can change the equilibrium strategies of *specific* games by introducing penalties (a form of taxation) on some actions of the players. Interesting applications include information security [13] and analyzing the TCP protocol. In the multi-agent systems community, Monderer and Tennenholtz proposed the notion of k implementation [8], whereby a principal can make payments to players (negative taxes) to incentivise players to choose certain outcomes. The setting for k -implementation is one of payments, in contrast to the present paper, and our use of Boolean games and logical objectives Υ is rather different. A related idea is discussed in [1], which considers how much compensation would have to be paid to players in a cooperative game in order for certain outcomes to become core stable.

We believe the results of the present paper strongly indicate that there are important and interesting theoretical and practical questions relating to non-incentive compatible taxation schemes. Future work might consider, for example: a complete characterisation of the conditions under which an objective Υ can be implemented in a game G ; consideration of the computation of taxation schemes τ for objectives Υ ; and the use of taxation schemes to incentivise behaviour in other settings, beyond the Boolean games considered in the present paper.

7. REFERENCES

- [1] Y. Bachrach, E. Elkind, R. Meir, D. Pasechnik, M. Zuckerman, J. Rothe, and J. S. Rosenschein. The cost of stability in coalitional games. In *Proceedings SAGT 2009*, 2009.
- [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] J. J. Cordes. Horizontal equity. In R. D. Ebel J. J. Cordes and J. G. Gravelle, editors, *The Encyclopedia of Taxation and Tax Policy*. Urban Institute Press, 1999.
- [4] 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.
- [5] E. Ephrati and J. S. Rosenschein. The Clarke tax as a consensus mechanism among automated agents. In *Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91)*, Anaheim, CA, 1991.
- [6] 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.
- [7] E. Maskin. The theory of implementation in Nash equilibrium: A survey. MIT Department of Economics Working Paper, 1983.
- [8] D. Monderer and M. Tennenholtz. k -implementation. *Journal of AI Research*, 21:37–62, 2004.
- [9] H. Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press: Cambridge, England, 1988.
- [10] N. Nisan and A. Ronen. Algorithmic mechanism design. In *Proceedings of the Thirty-first Annual ACM Symposium on the Theory of Computing (STOC-99)*, pages 129–140, May 1999.
- [11] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, editors. *Algorithmic Game Theory*. Cambridge University Press: Cambridge, England, 2007.
- [12] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. The MIT Press: Cambridge, MA, 1994.
- [13] W. Sun, X. Kong, D. He, and X. You. Information Security Game Analysis with Penalty Parameter. In *Electronic Commerce and Security, 2008 International Symposium on*, pages 453–456, 2008.
- [14] W. van der Hoek and M. Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence*, 164(1-2):81–119, May 2005.

Main Program – Full Papers

Robotics

Who Goes *There*?

Selecting a Robot to Reach a Goal Using Social Regret

Meytal Traub
The MAVERICK Group
Computer Science Dept.
Bar-Ilan University, Israel
meyaltra@gmail.com

Gal A. Kaminka
The MAVERICK Group
Computer Science Dept.
Bar-Ilan University, Israel
galk@cs.biu.ac.il

Noa Agmon
Computer Science Dept.
University of Texas at Austin
Austin, TX, USA
agmon@cs.utexas.edu

ABSTRACT

A common decision problem in multi-robot applications involves deciding on which robot, out of a group of N robots, should travel to a goal location, to carry out a task there. Trivially, this decision problem can be solved greedily, by selecting the robot with the shortest expected travel time. However, this ignores the inherent uncertainty in path traversal times; we may prefer a robot that is slower (but always takes the same time), over a robot that is expected to reach the goal faster, but on occasion takes a very long time to arrive. We make several contributions that address this challenge. First, we bring to bear economic decision-making theory, to distinguish between different selection policies, based on risk (risk averse, risk seeking, etc.). Second, we introduce *social regret* (the difference between the actual travel time by the selected robot, and the hypothetical time of other robots) to augment decision-making in practice. Then, we carry out experiments in simulation and with real robots, to demonstrate the usefulness of the selection procedures under real-world settings, and find that travel-time distributions have repeating characteristics.

Categories and Subject Descriptors

I.2.9 [Artificial Intelligence]: Robotics

General Terms

Algorithms, Economics, Performance, Experimentation

Keywords

Multi-Robot Systems, Decision-Making, Regret

1. INTRODUCTION

A common decision problem in multi-robot settings involves deciding on which robot, out of a group of N robots, should travel to a goal location, to carry out a task there. This decision repeats in many applications: in multi-robot exploration (e.g., deciding who should go to explore a new frontier), in package delivery robots (e.g., deciding who should go to pick up a package), and in other service robotics applications (e.g., in hospitals). In all of these, robots can plan a path to reach their destination, in an environment that is—for the most part—known to them. Thus, in principle, they can analytically predict their travel time to any location.

Cite as: Who Goes *There*? Selecting a Robot to Reach a Goal, Meytal Traub, Gal A. Kaminka and Noa Agmon, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Yolum, Tumer, Stone and Sonenberg (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 91–98.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Trivially, this decision problem can be solved greedily, by selecting the robot with the shortest predicted travel time [12], or using a market-based allocation scheme (see [4]). However, this ignores the inherent variance in the actual path traversal times, due both to motion and sensing errors, as well as multiple factors that affect a robot’s velocity (e.g., battery level, unknown obstacles). Solutions that have been proposed to address these challenges include using machine learning to better predict actual travel times under varying conditions [7, 14, 8], or other path-generation techniques that provide estimates [3, 2].

A common thread through previous work is that it focuses on scalar predictions; a single number that denotes the expected travel-time for each robot. Unfortunately, scalar predictions hide important information about the uncertainty in the predictions. In particular, a scalar denoting expected cost ignores information about the distribution of possible costs, best- and worst-case costs, etc. As a result, guarantees on the cost of task execution are not possible.

For instance, suppose that we must send one of two robots to a target location X . Robot A ’s path to X takes 100 seconds, through a free corridor. But if the corridor is busy with traffic (a rare occurrence), it may take up to 200. In contrast, robot B ’s travel time is always 150 seconds, through a specialized service way. Since the corridor is normally clear, we might choose robot A for the task. But if we wanted to absolutely guarantee delivery within 150 seconds, we would choose robot B . Note that if we only know the expected (i.e., mean) travel time, we cannot make the necessary distinction that allows this decision.

In this paper, we make several contributions that address the challenge involved in selecting a robot to go to a target location, given that each robot has a distribution over predicted travel times: First, we bring to bear economic decision theory that distinguishes between different selection policies, based on *risk*: risk averse, risk seeking, risk neutral, and bounded-risk selection. Second, we show that under some conditions, the selected robot may still not be a reasonable choice in practice. We thus introduce the use of *social regret* (the difference between the actual travel time by the selected robot, and the hypothetical time of another robot) to augment decision-making. Social regret is inspired by economic notions of regret, though the definitions differ.

Then, we carry out experiments in simulation and with real robots, to demonstrate the usefulness of the selection procedures under real-world settings. We empirically demonstrate that even under static conditions of the environment, when it is completely known to the robots, sensor and actuator errors leads to significant variance in the execution of path-following tasks. This variance leads to non-trivial distribution of costs, which in turn necessitates reasoning about the different optimization criteria when making the selection between robots. Finally, we show empirically that travel

time distributions have repeating characteristics (specifically, they fit extreme value distributions).

2. BACKGROUND

There have been several investigations that attempt to predict travel time or related costs. To the best of our knowledge, none addressed complete distributions.

Heero et al. [8] present a method for learning the shortest path in a partially-known environment by using a rectangular grid-based map. For each path they saved four parameters: number of re-plans, travel time, travel distance and deviation from the originally pre-planned path. Chaudhry [3] presented an algorithm for generating paths using matrix representation of the robot’s previous path traversals. New paths are created by applying transformations to the matrix, given the new path requirements. Both investigations use previous experiences to generate travel time predictions.

Sofman et al. [13] demonstrated an approach for learning Gaussian distributions associated with the local environments of the robot, to improve navigation and the travel speed. They use the models to generalize to new environments. They do not learn travel time distributions, and in any case as we show, travel time distributions are not Gaussian.

A related—though inverse—problem to ours is the problem of choosing a path, out of k possible paths, for a single robot to reach a goal location. Haigh and Veloso developed ROGUE [7]. It learns situation-dependent rules based on the success or failure in carrying out its tasks, and in particular, learns to take different paths depending on the time of day, expected use of the corridor, etc. ROUGE learns these situated-dependent cost predictions by examining the mean costs of travel for given locations. Thurn et al. developed MINERVA, an interactive tour guide robot for the Smithsonian museum [14]. It used POMDP methods to learn and plan its motion. The use of POMDP is similar to the risk-neutral policy, one of a number we present in this paper.

Our notion of regret is inspired by—but different than—notions of regret in economics. Economic regret were introduced by Bell [1] and by Loomes and Sugden [10], who concluded that people do not necessarily maximize their expected utility, but also consider the possible loss they are willing to accept from making a choice. They defined regret as a symmetric function with respect to two choices: choosing A rather than B minus the gain/loss from choosing B rather than A . In our case we calculate the regret with respect to all other choices, yet the comparison between the symmetric cases is done after the calculation (hence our regret function is asymmetric). Foster and Vohra [5] discussed regret in online decision-making, distinguishing between *internal* and *external* regret. Our definition of regret is similar to the definition of internal regret, however we evaluate with respect to the probabilities of costs, rather than on a limited history over time.

Market-based methods are sometimes used to assigning robots to tasks (e.g., a goal location to be reached; see [4] for a survey). In general, these methods rely on scalar cost estimates, and do not utilize information about travel cost distributions. However, they do address self-interest on the part of the robots, while in our work we assume robots are cooperative and truthful. Koenig et al. [9] uses a regret function, different from ours, to improve such auctions.

3. SELECTING A ROBOT

The problem is to select a robot R_i , out of a group of N robots R_1, \dots, R_N , to carry out a task, while minimizing the cost. We assume that each robot can estimate its cost of task execution with some discrete probability distribution over k cost values c_1, \dots, c_k .

Each robot R_i has a vector of size k , $\langle p_1^i, p_2^i, \dots, p_k^i \rangle$ such that p_j^i is the probability that the cost of task execution (travel time, in our case) by robot R_i is c_j and $\sum_{j=1}^k p_j^i = 1$ (note that p_j^i can be equal to 0). We use this discrete distribution formalization for simplicity, in lieu of the continuous distribution case which is more natural for estimated travel times. Note also that each robot’s travel time is an independent random variable, i.e., the probability of robot R_i having actual cost of c_j does not depend on the probability of some other robot having this or other cost.

We use the following running example throughout this section. Figure 1 shows three robots $\{R_1, R_2, R_3\}$. One of these robots is to be sent to explore a new frontier, F_1 , shown in the bottom right corner (circled). Each robot constructs a path (not shown) to the new location, and reports a distribution over estimated travel times. As we show in the experiments (Section 4.1), even in a completely static environment (let alone in dynamic environments), sensor and motion uncertainties cause some variance in this distribution.

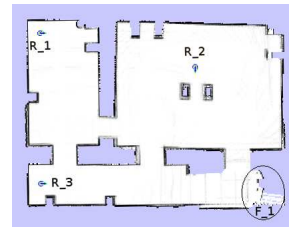


Figure 1: Three robots in exploration task. Map was generated using laser-based SLAM.

Suppose the travel time distributions reported by the 3 robots are as given in Table 1. Each row shows the distribution of a different robot, with different columns denoting different costs. The last column shows the mean (expected) cost for each robot. Given different decision objectives, we would choose different robots to go to F_1 . For instance, R_2 is most likely to reach F_1 faster (has a 87% chance of reaching F_1 in 86 seconds). But R_2 may also take up to 134 seconds for the same path. If we wanted to guarantee arrival within 2 minutes, we would choose R_3 .

	$c_1 =$ 86	$c_2 =$ 98	$c_3 =$ 110	$c_4 =$ 122	$c_5 =$ 134	$E(C)$
R_1	$p_1^1 =$ 0	$p_2^1 =$ 0.6	$p_3^1 =$ 0.23	$p_4^1 =$ 0.17	$p_5^1 =$ 0	104.84
R_2	$p_1^2 =$ 0.87	$p_2^2 =$ 0.03	$p_3^2 =$ 0	$p_4^2 =$ 0	$p_5^2 =$ 0.1	91.16
R_3	$p_1^3 =$ 0.6	$p_2^3 =$ 0.22	$p_3^3 =$ 0.1	$p_4^3 =$ 0.08	$p_5^3 =$ 0	93.92

Table 1: Possible cost distribution for $R_{1,2,3}$ for arriving to F_1 .

3.1 Risk-based selection

Choosing the robot $R_c \in \{R_1, \dots, R_N\}$ to perform the given task is dependent on a decision policy, which prefers robots—all else being equal—based on the risk involved. For instance, if we have a fixed amount of time to explore a given area, we may want to select a robot that will definitely reach its target within the time allotted. On the other hand, we may decide to take more risks, hoping to reach the target faster than expected.

Such decision policies are well known in economic decision theory. We distinguish four well-defined policies, and outline the selection algorithm for each:

1. Minimize the expected travel time (risk neutral selection).
2. Minimize the expected maximal travel time (risk averse selection).

3. maximize the expected minimal travel time (risk seeking selection).
4. Bound the travel time by a constant A (bounded risk).

Risk-Neutral Selection. Risk-neutral selection implies that we select the robot that minimizes the expected (mean) travel time. To do this, we compute the mean of every robot's distribution, and choose the robot whose mean is minimal

$$\text{MinExp}_C = \operatorname{argmin}_{1 \leq a \leq N} \left\{ \sum_{i=1}^k p_i^a c_i \right\}$$

where in case of a tie, we choose arbitrarily.

Risk-Averse Selection. In some cases we want to make sure that the worst-case scenario is addressed first, and that we have an absolute guarantee that the task will be carried out within a given amount of time. To do this, we need to look at the robots whose greatest time of arrival is minimal. Of course, the probability of actually taking this long time must also be taken into account. Thus what we want is to find the robot which minimizes the expected maximal cost. This is done in Algorithm 1.

Algorithm 1 MinExpMaxCost(R)

Require: $C = \{c_1, c_2, \dots, c_k\}, R = \{R_1, R_2, \dots, R_n\}$
 $v \leftarrow k$
 $\text{Robots}_{list} \leftarrow \{R_1, R_2, \dots, R_n\}$
while $\exists p_v^i = p_v^h, R_j, R_h \in \text{Robots}_{list}$ **do**
 $\text{Robots}_{list} \leftarrow \operatorname{argmin}_{R_i \in \text{Robots}_{list}} \{p_v^i\}$
 $v \leftarrow v - 1$
return : $\operatorname{argmin}_{R_i \in \text{Robots}_{list}} \{p_v^i\}$

Note that ties can be broken in different ways. For instance, we can choose the robot with the lower expected time among those that are returned.

We use Table 1 to illustrate. The algorithm creates a list of all the robots that available to execute the task $\{R_1, R_2, R_3\}$, and starts the run with the highest cost (134). It looks for two robots with the same probability to arrive the goal in cost 134. In this example R_1 and R_3 have the same probability ($p_5^1 = p_5^3 = 0$), so the loop will be entered. The algorithm choose the robots with the minimal probability to execute the task in cost 134 by argmin . By doing it, all the robots with probability higher than 0 will be removed from the list, i.e. robot R_2 . The algorithm then examines the next highest cost, 122. While looking on the remaining robots $\{R_1, R_3\}$, their probability to arrive the target is different, therefor the loop will not be entered and the robot with the lowest probability to use this cost will be returned: R_3 ($p_4^1 = 0.17 > p_4^3 = 0.08$).

Risk Seeking Selection. The opposite policy to being risk averse is to be risk seeking; to hope for the best possible travel time of any of the robots. Here the selection is exactly the inverse of the above: We select the robot that maximizes the expected minimal cost. Algorithm 2 is thus the inversion of Algorithm 1.

We again use Table 1 to illustrate. The algorithm starts the run with the lowest cost, i.e. 86. It looks for two robots with the same probability to arrive the goal in cost 86. In this example, there is no two such robots, and so it does not enter the loop and return the robot with the highest probability to arrive the goal in this cost, R_2 ($p_1^1 = 0 < p_1^3 = 0.6 < p_1^2 = 0.87$).

Algorithm 2 MaxExpMinCost(R)

Require: $C = \{c_1, c_2, \dots, c_k\}, R = \{R_1, R_2, \dots, R_n\}$
 $v \leftarrow 1$
 $\text{Robots}_{list} \leftarrow \{R_1, R_2, \dots, R_n\}$
while $\exists p_v^j = p_v^h, R_j, R_h \in \text{Robots}_{list}$ **do**
 $\text{Robots}_{list} \leftarrow \operatorname{argmax}_{R_i \in \text{Robots}_{list}} \{p_v^i\}$
 $v \leftarrow v + 1$
return : $\operatorname{argmax}_{R_i \in \text{Robots}_{list}} \{p_v^i\}$

Bounded-Risk Selection. Finally, we may want to choose the robot that maximizes the probability of reaching the target within some limited amount of time. This is different from guaranteeing arrival within this time; it would still be possible that in the worst case, travel time will be longer. Nevertheless, we want to improve its chances of success within the time allotted.

Suppose we are given a time limit T . We can then calculate for each robot the cumulative probability that its travel time be smaller than T , and choose the robot that maximizes this probability.

For each robot R_a , we will calculate the following probability:

$$P[C \leq T] = \sum_{c_i \leq T, c_i \in C} p_i^a c_i$$

We will choose the robot that maximize the result of this equation. Note, that if only one robot have distribution of cost bellow the constant, then it will be chosen with probability of 1. If there is more than one robot that fits this, then we can select based on any of the other criteria (e.g., the best risk-seeking robot out of the candidates that fit the bound T).

3.2 Regretting the Selection

Despite the economic elegance of the selection policies described above, choosing the robot according to the risk type will not always give us a reasonable selection in practice. To see this, consider the following case (Table 2). Here, we apply the risk-averse policy, and select R_2 : It is guaranteed to reach the goal in 199 seconds. However, unless this risk-averseness is somehow extremely strict, R_1 would have been a more reasonable choice: 90% of the time it would have reached the goal in 1 second. And even when it fails, it would do it in 200 seconds, a mere 1 second more than R_2 .

	$c_1 = 1$	$c_2 = 199$	$c_3 = 200$	$E(C)$	E_{SoR}
R_1	$p_1^1 = 0.9$	$p_2^1 = 0$	$p_3^1 = 0.1$	20.9	0.1
R_2	$p_1^2 = 0$	$p_2^2 = 1$	$p_3^2 = 0$	199	178.2

Table 2: Robots distributions of costs to arrive at a goal. Expected cost and expected SoR are shown. Selecting R_1 over R_2 makes sense in practice.

Note that this is not always the case: It depends very much on the values c_i . If the c_i would have been 1, 2, 200 rather than 1, 199, 200, our deliberation would not have reached the same conclusion, and the selection of R_2 would have held.

To conduct this deliberation formally, we define the *social regret* function, which measures, intuitively, the post-hoc payment (in travel time) that we make, given the selected robot.

Social Regret SoR is defined as the difference between the *actual* cost c_r of the task executed by robot R_a and the minimal cost of task execution in case some other robot would have executed the task in lower cost. In other words, looking at it from the team's

perspective: How bad did the team do by choosing robot R_a to perform the task, given R_a 's actual cost was c_r . Formally, SoR of robot R_a executing a task with actual cost c_r is $SoR(R_a, c_r) = \max_{j \neq i} (c_r - c_j, r > j)$.

Since we do not know SoR for any specific selection (it is by definition hypothetical), we compute the *expected SoR* for each robot R_a , given all other robots, and all possible outcomes. The expected social regret from choosing R_a , $E_{SoR}(R_a)$, is the probability that some other robot will execute the task with lower cost multiplied by the difference between the costs. We denote the probability that the actual *minimal* cost of task execution by some robot other than R_a is c_i by $PM_i(R_a)$. Note that by *minimal* cost we mean that there is no other robot that executed the task with cost $c_j, j < i$, and that at least one robot executed the task with cost c_i . Therefore, $E_{SoR}(R_a) = p_1^a \times 0 + p_2^a \times PM_1(R_a)(c_2 - c_1) + p_3^a \times [PM_1(R_a)(c_3 - c_1) + PM_2(R_a)(c_3 - c_2)] + \dots$, and formally

$$E_{SoR}(R_a) = \sum_{i=2}^k p_i^a \times \sum_{j=1}^{i-1} PM_j(R_a)(c_i - c_j)$$

In order to complete the definition, it is necessary to determine $PM_j(R_a)$, i.e., the probability that some robot $R_o, 1 \leq o \leq N, o \neq a$ will have minimal cost of c_j . This is the probability that all robots have minimal cost higher than c_{j-1} minus the probability that all robots have minimal cost higher than c_j , i.e., $E_{SoR}(R_a) =$

$$\sum_{i=2}^k p_i^a \left\{ \sum_{j=1}^{i-1} (c_i - c_j) \left[\prod_{h=1}^{N, h \neq a} \left(\sum_{l=j}^k p_l^h \right) - \prod_{h=1}^{N, h \neq a} \left(\sum_{l=j+1}^k p_l^h \right) \right] \right\}$$

3.2.1 When Should We Overrule The Selection?

Intuitively, $E_{SoR}(R_a)$ measures the potential cost of selecting R_a to carry out a task, given the estimated costs of its peers. Suppose that we have two robots R_i and R_j . What we want, is to compare the difference in the expected SoR of the two robots, to the gain from choosing one over the other. If this gain is smaller than the difference in expected SoR , then we should consider switching between them.

To illustrate, suppose R_i has been selected by some policy, and has a predicted travel time c_i (this is, for instance, its maximal time). Suppose we want to consider switching to a different robot R_j , with predicted cost c_j . In order to compute the profit from switching two robots we will calculate the distance between the expected SoR of R_i and R_j , $E_{SoR}(R_i) - E_{SoR}(R_j)$. We compare this value to the difference in costs between R_i and R_j , which is $(c_j - c_i)$, using the following function.

The Switch function SwF is defined as follows:

$$SwF = \begin{cases} 1 & \text{if } (E_{SoR}(R_i) - E_{SoR}(R_j)) > (c_j - c_i) \\ 0 & \text{otherwise} \end{cases}$$

If the SwF is 1, the social regret of using R_i is greater than the expected gain of using it, and we should consider switching our selection to R_j instead. We examine this in different selection policies below.

Minimize the expected maximal cost. Table 2 above describes the cost distributions for two robots, R_1 and R_2 . As previously discussed, strict risk-averse policy would select R_2 for the task, since it is guaranteed to reach the target in 199 seconds. However, by risking just one additional second, we actually have much better average performance if we choose R_1 .

SwF identifies this opportunity. The difference between c_3 (R_1 's cost) and c_2 (R_2 's cost) is 1, while the distance between the

$E_{SoR}(R_2)$ and $E_{SoR}(R_1)$ is 178.1. Thus SwF is 1, and we should consider switching our selection to the other robot.

Switching in the case of a bounded risk. Using a bounded-risk policy, we normally select the path that is most likely to carry out the task within the time allotted. But by bounding the cost, we are not bounding the regret function. In other words, choosing the best robot given the bound T , does not reduce our expected SoR for the bounded cost, and we can still choose to switch based on SwF .

For example, table 3.2.1 shows distributions of costs of two robots, R_1 and R_2 . A bound of $T = 7$, yields selection R_1 (with cumulative likelihood 0.2), over R_2 (cumulative likelihood 0). But the expected SoR of R_1 is much higher than the expected SoR of R_2 . Indeed, using the SwF we might consider to change the constant T to be higher. By changing the constant T from 7 to 10, R_2 will have higher probability than R_1 to execute the task under the new bound. We will pay 3 in the bound but gain 70.6 in the expected regret ($E_{SoR}(R_1) - E_{SoR}(R_2)$). The SwF will be 1 ($70.6 > 3$).

	$c_1 = 1$	$c_2 = 5$	$c_3 = 10$	$c_4 = 100$	E_{SoR}
R_1	$p_1^1 = 0.1$	$p_2^1 = 0.1$	$p_3^1 = 0$	$p_4^1 = 0.8$	72
R_2	$p_1^2 = 0$	$p_2^2 = 0$	$p_3^2 = 1$	$p_4^2 = 0$	1.4

Table 3: The bounded cost does not minimize the expected SoR . When should we replace

3.2.2 Minimal Expected Cost is Safe Selection

For one of the policies we introduced, it turns out that we do not need to consider regret. We prove that by minimizing the expected cost, the expected social regret function, SoR , is minimized as well, and thus we would not want to switch to a different robot.

First, we show in Lemma 1 that for a pair of robots, minimizing the expected cost leads to minimization of the expected SoR . We then complete the proof for N robots in Theorem 2.

LEMMA 1. *For two robots R_1 and R_2 with discrete probability distribution $\{p_1^1, p_2^1, \dots, p_k^1\}$ and $\{p_1^2, p_2^2, \dots, p_k^2\}$ (respectively) over possible costs c_1, \dots, c_k of a given task, $c_i < c_{i+1}$, if the robot minimizing the expected cost of the task execution is chosen, then the expected social regret function SoR is minimized.*

PROOF. Assume, without loss of generality, that robot R_1 minimizes the expected cost of the task execution, i.e., $\sum_{i=1}^k p_i^1 c_i < \sum_{i=1}^k p_i^2 c_i$, i.e., $\sum_{i=1}^k c_i p_i^2 - c_i p_i^1 > 0$. We therefore need to show that $E_{SoR}(R_2) - E_{SoR}(R_1) > 0$.

First, note that since $N = 2$ then $PM_j(R_1) = p_j^2$, and similarly $PM_j(R_2) = p_j^1$, therefore $E_{SoR}(R_2) = \sum_{i=2}^k p_i^2 \times \sum_{j=1}^{i-1} p_j^1 (c_i - c_j)$ (similarly $E_{SoR}(R_1) = \sum_{i=2}^k p_i^1 \times \sum_{j=1}^{i-1} p_j^2 (c_i - c_j)$).

By opening the formula of $E_{SoR}(R_2)$ we get that $E_{SoR}(R_2) = p_2^2 \{p_1^1 (c_2 - c_1)\} + p_3^2 \{p_1^1 (c_3 - c_1) + p_2^1 (c_3 - c_2)\} + p_4^2 \{p_1^1 (c_4 - c_1) + p_2^1 (c_4 - c_2) + p_3^1 (c_4 - c_3)\} + \dots = c_1 \{-p_1^1 p_2^2 - p_1^1 p_3^2 - p_1^1 p_4^2 - \dots\} + c_2 \{p_1^1 p_2^2 - p_2^1 p_3^2 - p_2^1 p_4^2 - \dots\} + c_3 \{p_1^1 p_3^2 + p_2^1 p_3^2 - p_3^1 p_4^2 - \dots\} + \dots$ Therefore $E_{SoR}(R_2)$ can be represented as $\sum_{j=1}^k c_j [\sum_{i=1}^j p_j^2 p_i^1 - \sum_{i=j+1}^k p_i^2 p_j^1] = \sum_{j=1}^k c_j [\sum_{i=1}^j p_j^2 p_i^1 - p_j^1 (1 - \sum_{i=1}^j p_i^2)] = -\sum_{j=1}^k c_j p_j^1 + \sum_{j=1}^k c_j [\sum_{i=1}^j (p_j^2 p_i^1 + p_j^1 p_i^2)]$. Similarly, $E_{SoR}(R_1) = -\sum_{j=1}^k c_j p_j^2 + \sum_{j=1}^k c_j [\sum_{i=1}^j (p_j^1 p_i^2 + p_j^2 p_i^1)]$, therefore $E_{SoR}(R_2) - E_{SoR}(R_1) = \sum_{j=1}^k (c_j p_j^2 - c_j p_j^1) +$

$\sum_{j=1}^k c_j [\sum_{i=1}^j (p_j^2 p_i^1 + p_j^1 p_i^2) - (p_j^1 p_i^2 + p_j^2 p_i^1)] = \sum_{j=1}^k (c_j p_j^2 - c_j p_j^1)$ and by initial assumption this is greater than 0, which completes the proof. \square

THEOREM 2. *Given a team of N robots $\{R_1, \dots, R_N\}$ each with a discrete probability distribution over possible costs v_1, \dots, v_k for a given task, if we choose a robot R_c that minimizes the expected cost for the task, then the expected social regret function SoR is minimized.*

PROOF. (Sketch). By induction on the number of robots N . Assume, without loss of generality, that robot R_1 minimizes the expected cost of the task execution. For the induction base case $N = 2$, the theorem holds based on the Lemma 1. For the inductive step, suppose the theorem holds for $N - 1$ robots, where $N \geq 3$. We will show that if we choose R_c that minimizes the expected cost, then $E_{SoR}(R_N)$ is minimized.

Let $\{R_1, \dots, R_N\}$ be a team of N robots. We will assume, without loss of generality that R_1 minimizes the executed cost of the task execution. Therefore in particular, R_1 minimizes the expected cost for all the possible couples of robots in the environment, i.e., for any given pair of robots (R_1, R_i) where $1 \leq i \leq N$, the selection of R_1 results in a minimal cost compared to the selection of R_i . We will prove by contradiction that R_1 minimizes the expected SoR for N robots.

We assume for contradiction that exist robot R_i , where R_1 's expected cost is smaller than R_i 's expected cost ($E_C(R_1) < E_C(R_N)$). But according to lemma 1, Let R_1 and R_i be a team of two robots, if $E_C(R_1) < E_C(R_i)$ then the expected social regret function SoR is minimized, R_1 's expected SoR is bigger than R_i 's expected SoR ($E_{SoR}(R_1) > E_{SoR}(R_i)$). In contradiction to the assumption. Therefore, the theorem holds for N robots, $N \geq 2$. \square

Table 2 gives travel times distributions of two robots to a goal. As it shows in the table, if we will choose robot R_1 by minimizing its expected cost, we will minimize the E_{SoR} as well.

3.2.3 A Short-Cut to Determining SwF

The computation of E_{SoR} for each robot, which is necessary whenever we select robots based on a policy different from risk-neutral selection, is tedious, and potentially time-consuming if the distribution's domains are large, or there are many robots.

Thankfully, it turns out that we do not need to compute E_{SoR} directly. To compute SwF , we want the difference $E_{SoR}(R_i) - E_{SoR}(R_j)$ for the two robots R_i, R_j . It turns out that a corollary of Lemma 1 is that this difference is exactly the difference in expected costs of the two robots, which is much easier to compute:

COROLLARY 3. *From Lemma 1, it follows that the difference between the expected costs of any two robots in a given team of N robots $\{R_1, \dots, R_N\}$ (each with a discrete probability distribution over possible costs c_1, \dots, c_k) is equal to the distance between the expected SoR of the same robots.*

PROOF. Omitted for lack of space. \square

4. PATH TRAVEL IN PRACTICE

We experimented with simulated and physical robots, to examine the travel time distributions in practice. We use the results to demonstrate in Section 4.1 that even under ideal conditions, robots do indeed have variance in the time that it takes them to travel a given path, and that this variance needs to be taken into account as described above. In Section 4.3 we show that the travel time distributions have distinctive shapes, and in general fit the Generalized

Extreme Value family of distributions, and thus the distributions can be estimated in principle.

4.1 Experiments with Robots

We used our laboratory as the environment for the experiments. First, we used a popular open-source laser-based SLAM package, GMapping [6], to allow the robots to construct a map of the environment. The results of the exploration and mapping process were used as the basis for the experiments; this is to make sure that all path-planning and movements were carried out using a map with realistic quality. For path planning, we used A^* with a fixed 4-neighbor grid laid out over the map. If the robot discovered an unknown obstacle on the way, it tried to go around the obstacle until a timeout occurred, in this case a new path was planned from the current location to the goal, given the new information about the discovered obstacle.

Physical Robot Experiments. We utilized the RV-400 differential-drive robots (see Figure 2) for experiments in our lab. The RV-400 was equipped with a Hokuyo UTM-30LX laser, with nominal range of 30m (though in practice effective range was slightly smaller). The RV-400 robot has an approximate size 40×40 (width, length), and so this was used as the grid cell-size. We kept the environment static, with no obstacles or other changes to the environment that are unknown to the robot.

Figure 3 shows the environment used for the experiments, as mapped by the robot. We tested three paths: A short 6.4m path (2 to 3), with a narrow pass; an 8m path (1 to 2), through open space; and a 14.6m path which combined both (1 to 3). We measured the travel time in each of these paths 10 times.



Figure 2: RV400 robot.

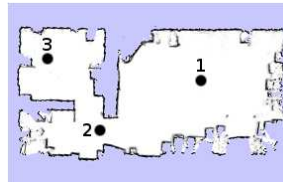


Figure 3: The mapped lab used in the robotics experiments.

Figures 4(a), 4(b), and 4(c) show the distribution, in histogram form, of travel time that were measured in these experiments, for the 6.4m, 8m, and 14m paths. For each of the settings, the planned path was identical, and the environment kept strictly static. The associated figure shows the path traversal time (X axis) versus its probability (Y Axis). Despite these ideal conditions, the robot took varying amount of time getting to the target locations. This variance is caused because of inaccuracies in the movement and sensing, which lead to actual execution of the path to differ between runs. In addition, changes to battery power also affect the robots linear and angular velocities. Indeed, Figure 4(b) does not include four data points that were removed from the data, because in their associated runs the robot operated with a faulty battery, and was almost twice as slow as in the other runs.

Simulation Experiments. We also conducted experiments in simulation, where we scaled up the number and complexity of the paths. We utilized

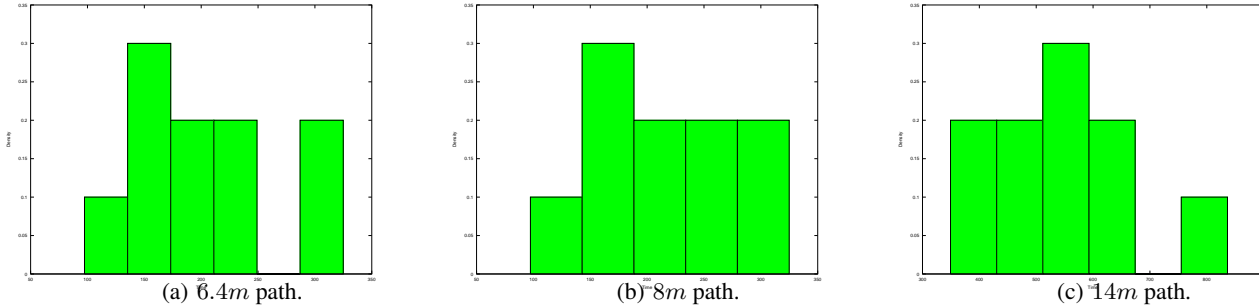


Figure 4: RV-400 Travel Time Distributions.

the Webots 3D physics-based robotics simulator [11] to create the virtual world which the robots mapped and navigated as part of the experiments (see Fig. 5 for the resulting map). Webots has high fidelity, and models realistic sensor and motion errors, as we demonstrate below. In the simulation experiments, we simulated three RV-400 robots and their Hokuyu lasers. The openings between the rooms are doors which were open or close according to the evaluated criteria. Minor obstacles (boxes to be bypassed) are not shown. The doorway between the rooms is $1.2m$ wide.

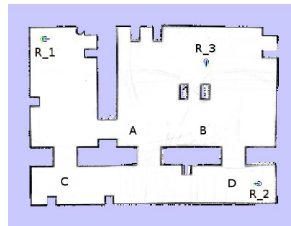


Figure 5: The mapped simulated environment.

The following configurations were used in the simulation experiments: From every robot location, to targets location A, B, C (9 combinations), and robots R_1, R_3 to target location D . we tested 4 obstacle settings: (i) static world (i.e., conforming to the map); (ii) with an unknown obstacle (a box placed on the planned path, that can be avoided and bypassed); (iii) an unexpected closed door blocking the original path (if the path was through an opening); and (iv) two unexpected closed doors blocking the original path, then a re-planned path. Each of the configuration (11 initial-target location pairs, 4 obstacle settings) was repeated 30 times. In all the experiments the robot had a path to the last target.

A small subset of the results from the simulation experiments are shown in Figures 6(a)–6(c). These are the results for one robot R_1 , and for a single target point A (results for point B are shown later; other robots and points omitted for lack of space). Our intent is to demonstrate the variance that exists even under idealized simulated conditions.

Figure 6(a) shows the distribution of traversal times of R_1 for arriving at target A in a static world. As seen in the figure, even for a static world, and even under the relative noise-free world of simulation, there is variance in traversal time, due to motion and sensing uncertainties.

Of course, when choosing a robot for executing a task the world cannot typically be assumed to be static. These increase the variance in the actual travel times. Figure 6(b) shows the wider distribution of traversal times when an obstacle was added to the path of the robot, in 50% of 60 cases (the X axis scale is 80 to 350). This obstacle could be locally avoided (bypassed), and thus only a minor change was required to the pre-planned path. Note, that all the distribution of the static environment become a part of the first bin of the new distribution. Figure 6(c) shows the even wider distribution

when we also take into account a door that was closed in a third of 90 cases, and which blocked the original path. This requires a new path to be planned and executed from the point where the closed door was discovered, to the target location.

The results above are similar to the distributions collected for the other experiment configurations, i.e., for other robots and other target locations. In all cases, even for paths that involve very few heading changes, and no obstacles or narrow passages, we see distributions that require reasoning about which robots to select, given the decision-maker’s policy towards risk.

4.2 Selection Based on Experiment Data

We use the collected data to execute the decision-making policies described earlier, in both the simulated and physical world. To do this, we discretized the collected data into bins of approximately 25 seconds, and chose the robots according to the different decision policies.

Simulation Experiments. Robots $\{R_1, R_2, R_3\}$ compete on reaching targets A, B, C , and robots $\{R_1, R_3\}$ on target D . Table 4 shows the chosen robot using each of the decision policies.

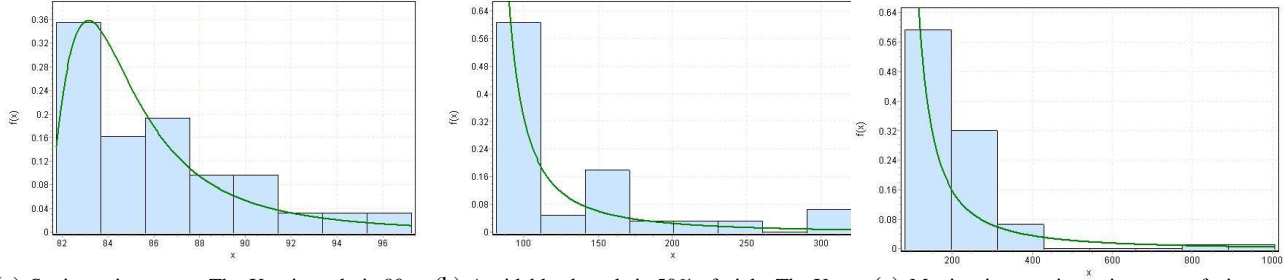
	$MinExp_C$ (risk-neutral)	$MinExpMax_C$ (risk-averse)	$MaxExpMin_C$ (risk-seeking)
A	R_3	R_3	R_1
B	R_3	R_3	R_2
C	R_2	R_2	R_2
D	R_1	R_3	R_3

Table 4: Selected robots for targets, according to each policy.

We find that indeed, the selected robot is not always the closest one to the target. For instance, R_3 is closest to point A . But when selecting a risk-seeking policy, R_1 is chosen. Likewise, R_3 is closest to point B , and yet R_2 is selected when a risk-seeking policy. R_3 is also closer to D , yet R_1 is selected in the risk-neutral policy. This is a direct result of the uncertainty inherent in the robots’ movements.

We note that the selected robots for points $\{A, B, C\}$ in the risk-averse $MinExpMax_C$ criteria were the same as the robots with the minimal expected cost. This is because the robots were in the same rooms with the target locations, and thus the closing and opening of doors—which would otherwise create large worst case travel times (and therefore large expected maximal times)—did not affect the ability of the robots to reach these targets.

Table 5 shows a case where an overruling of the selected robot is recommended by the SwF function. When selecting which of the



(a) Static environment. The X axis scale is 80 to 100. (b) Avoidable obstacle in 50% of trials. The X axis scale is 80 to 350. (c) Moving in a static environment, facing occasional avoidable obstacles, and sometimes needing to re-plan a path. The X axis scale is 80 to 1050.

Figure 6: R_1 Travel Time Distributions.

robots should reach target A through a risk-seeking policy, both robots R_1, R_2 have a minimal cost of 88. However, robot R_1 is chosen because its probability for this cost is a bit higher.

	p_{88}	E_{SoR}
R_1	0.438679	84.5507
R_2	0.433333	49.2834
R_3	0	29.0444

Table 5: The robots expected minimal cost, and expected SoR for the minimal cost of 88, while competing on point A .

But looking at the E_{SoR} of the robots, it is clear that R_2 has lower expected regret than R_1 . Plugging these values into the SwF function yields the following:

$$\begin{aligned}
 (E_{SoR}(R_1) - E_{SoR}(R_2)) &= 84.5507 - 49.2834 & (1) \\
 &= 35.2673 & (2) \\
 &> 0 & (3) \\
 &= 88 - 88 & (4) \\
 &= \min_C(R_2) - \min_C(R_1) & (5)
 \end{aligned}$$

In this case, SwF returns 1, and we should consider selecting R_2 despite its slightly higher expected minimal traversal time.

Physical Robot Experiments. We utilized the RV-400 data in similar experiments. Abstracting away from the map, we used the distributions for traversal times of 6.4m, 8m and 14.6m paths, for three robots: RV_1 positioned 6.4m away from a target point, RV_2 positioned 8m away from the same point, and RV_3 which is positioned 14.6m away. Table 6 shows the chosen robot in each of the decision policies.

$MinExp_C$ (risk-neutral)	$MinExpMax_C$ (risk-averse)	$MaxExpMin_C$ (risk-seeking)
RV_1	RV_2	RV_1

Table 6: Selected physical robot, according to each policy.

The results show that in the physical world as well, the closest robot is not always the robot to choose. Due to the narrow pass in the 6.4m path, the worst case travel time for RV_1 was worse (though less likely) than the worst case of RV_2 (which traveled 8m through open space).

4.3 Parametric travel time distributions

The experiments conducted reveal repeating characteristics of the emerging distributions, in particular their sharp lower bound

and long tail. This is a result of having a clear lower bound on path traversal time (there’s a limit as to how quickly a path can be traversed), and the increasingly rare (but still occurring) long arrival times, due to getting stuck by unforeseen obstacles, decreasing battery levels, etc. On such occasions, robots would re-plan their path several times on the way to the goal, and would sometimes need to traverse long distances to bypass a closed door.

We thus hypothesized that in fact known (parametrized) heavy-tailed continuous distributions may fit the data, allowing for improved prediction. We began experimentally, by fitting familiar distributions to the data, and using the Kolmogorov-Smirnov and Anderson-Darling fitness tests to determine the best-fitting distributions.

The fitness results for the best three distributions are shown in Table 7. The table shows the average matching functions, for all the paths that were followed, for the top three matching functions that were found.

	Gen. Logistic	Gen. Extreme Value	Frechet ($3P$)
Kolmogorov-Smirnov	0.132	0.135	0.136
Anderson-Darling	1.051	1.512	0.69

Table 7: The average fitness of the top three matching distributions using Kolmogorov-Smirnov & Anderson-Darling tests.

The three best-fitting functions were found to be the General Log-Logistic (also called the 3-parameter Log-Logistic distribution), the General Extreme Value, a limit distribution of the maximum of a sequence of independent random variables which are identically distributed. and Frechet ($3P$), a special case of the General Extreme Value distribution. The table shows that The General Log-Logistic distribution has the best average fitness using Kolmogorov-Smirnov test, and Frechet ($3P$) has the best average fitness using Anderson-Darling test. Both of them, however, are strongly related (special cases) of the General Extreme Value distribution. Figures 6(a), 6(b), 6(c) show a curve which is the best-fit Log-Logistic continuous probability distribution fitting the simulation experiments data. In addition, Figure 7 shows the travel times distribution of robot R_1 , traveling to point B in a static environment, 132 times. As seen is the figure, although the number of experiments grow the log-logistic distribution is a good fit.

We focused on the General Log-Logistic distribution. It has three parameters: shape, scale and shift. The shift parameter was found to be almost perfectly linearly correlated with the the minimal travel time of each one on the paths that were traveled. Figure 8 shows the relation between the minimal execution time of

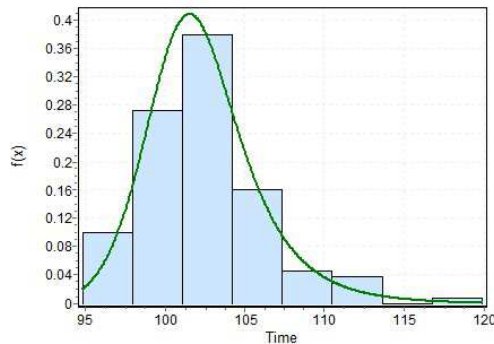


Figure 7: Distribution of R_1 's travel times to point B in a static environment, over 132 path following experiments. The line shows the fitted log-logistic distribution.

all the paths that were traveled, and the shift parameter of the General Log-Logistic distribution that was fitted to the histogram of the path travel times. It is clear from the figure that there exist a direct relation between the two.

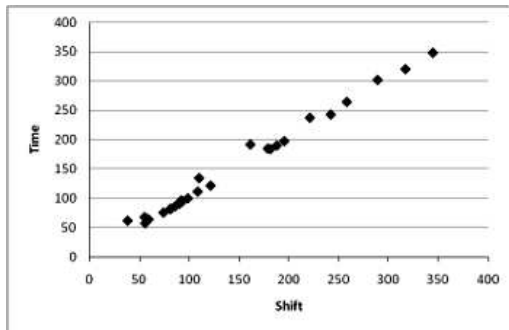


Figure 8: Measured minimal travel time versus fitted shift.

Looking on the other parameters, we found that the shape parameter was quite steady on values between 0.89304 to 3.3684 while its declaration is $(-\infty, \infty)$. We did not find a consistent value for the scale parameter.

5. DISCUSSION AND FUTURE WORK

The techniques developed in this paper allow robot selection that maintains bounds and guarantees as to travel times, even under uncertainty. We showed that even under static environment conditions, it takes the robots varying amount of time getting to a target location. Due to this variance, choosing a robot to preform a task cannot be done based on greedy selection (shortest path).

Thus, we introduced a decision making technique, inspired by economic decision theory, to distinguish between different policies based on risk. The experiments in simulated and physical robots demonstrated that different robots were chosen according to the different policies because of time travel variance: And indeed sometimes the closest robot is not the one to be selected, given the decision-making policy. Furthermore, we have shown that under some conditions, choosing the robot according to the selection policies will not always give a reasonable selection in practice. We defined the social regret function SoR which measure the cost of choosing specific robot over all other robot, and allow us to evaluate the gain from switching the chosen robot to a robot that will preform better. In our future work we plan to expand this techniques for allocating teams of N robots to K tasks.

While examining the experiments' data, we found that the data distributions had a good fit to the family of General Extreme Value distributions, and specifically to the General Log-Logistic distribution. We plan to explore this fit and learn to predict the parameters of the distributions for future real world paths and obstacles.

Acknowledgments. We thank the anonymous reviewers, and ISF grant #1357/07. As always, thanks to K. Ushi.

6. REFERENCES

- [1] D. E. Bell. Regret in decision making under uncertainty. *Operations Research*, 30(5):961–981, 1982.
- [2] J. Bobrow. Optimal robot path planning using the minimum-time criterion. *IEEE Journal of Robotics and Automation*, 4(4):443–450, 1988.
- [3] A. Chaudhry. Path generation using matrix representations of previous robot state data. In *2006 45th IEEE Conference on Decision and Control*, pages 6790–6795, 2006.
- [4] M. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: A survey and analysis. *Proceedings of the IEEE*, 94(7):1257–1270, 2006.
- [5] D. Foster and R. Vohra. Regret in the on-line decision problem. *Games and Economic Behavior*, 29(1–2):7–35, 1999.
- [6] G. Grisettiyz, C. Stachniss, and W. Burgard. Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling. In *ICRA-05*, pages 2432–2437, 2005.
- [7] K. Z. Haigh and M. M. Veloso. Planning, execution and learning in a robotic agent. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 120–127. AAAI Press, 1998.
- [8] K. Heero, J. Willemsen, A. Aabloo, and M. Kruusmaa. Robots find a better way: A learning method for mobile robot navigation in partially unknown environments, 2004.
- [9] S. Koenig, X. Zheng, C. Tovey, R. Borie, P. Kilby, V. Markakis, and P. Keskinocak. Agent coordination with regret clearing. In *Proceedings of the Twenty-Third National Conference on Artificial Intelligence (AAAI-08)*, pages 101–107, 2008.
- [10] G. Loomes and R. Sugden. Regret theory: An alternative theory of rational choice under uncertainty. *The Economic Journal*, 92(368):805–824, 1982.
- [11] O. Michel. WebotsTM: Professional mobile robot simulation. *CoRR*, abs/cs/0412052, 2004.
- [12] R. Simmons, D. Apfelbaum, W. Burgard, M. Fox, D. an Moors, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *AAAI-00*, 2000.
- [13] B. Sofman, E. Lin, J. A. Bagnell, J. Cole, N. Vandapel, and A. Stentz. Improving robot navigation through self-supervised online learning. *Journal of Field Robotics*, 23(11-12):1059–1075, 2006.
- [14] S. Thrun, M. Beetz, M. Bennewitz, W. Burgard, A. B. Cremers, F. Dellaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, et al. Probabilistic algorithms and the interactive museum tour-guide robot Minerva. *The International Journal of Robotics Research*, 19(11):972–999, 2000.

Exploration strategies based on Multi-Criteria Decision Making for search and rescue autonomous robots

Nicola Basilico
Politecnico di Milano, Italy
basilico@elet.polimi.it

Francesco Amigoni
Politecnico di Milano, Italy
amigoni@elet.polimi.it

ABSTRACT

Autonomous mobile robots are considered a valuable technology for search and rescue applications, where an initially unknown environment has to be explored to locate human victims. In this scenario, robots exploit exploration strategies to autonomously move around the environment. Most of the strategies proposed in literature are based on the idea of evaluating a number of candidate locations according to *ad hoc* utility functions that combine different criteria. In this paper, we show some of the advantages of using a more theoretically-grounded approach, based on Multi-Criteria Decision Making (MCDM), to define exploration strategies for robots employed in search and rescue applications. We implemented some MCDM-based exploration strategies within an existing robot controller and we experimentally evaluated their performance in a simulated environment.

Categories and Subject Descriptors

I.2.9 [Artificial Intelligence]: Robotics—Autonomous Vehicles

General Terms

Algorithms

Keywords

Mapping, search and rescue, exploration strategies

1. INTRODUCTION

In search and rescue with autonomous mobile robots, an initially unknown environment has to be explored and searched for human victims [4]. *Exploration strategies* that drive the robots around the partially known environment on the basis of the available knowledge are fundamental for achieving an effective behavior. The mainstream approach for developing exploration strategies is based on the idea of incrementally exploring the environment by evaluating a number of candidate observation locations according to an utility function and by selecting, at each step, the next best observation location. Exploration strategies differ in the utility functions they use to evaluate candidate locations. Although in multirobot exploration the evaluation of candidate observation locations is closely related to their coordinated allocations to the

Cite as: Exploration strategies based on Multi-Criteria Decision Making for search and rescue autonomous robots, N. Basilico and F. Amigoni, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 99-106.
Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

available robots, in this paper we focus only on evaluation of candidate observation locations. In systems proposed in literature, this evaluation is performed using utility functions that aggregate multiple criteria measuring different aspects of the locations and that are rarely based on a theoretical ground.

In this paper, we apply a decision-theoretical tool, called *Multi-Criteria Decision Making (MCDM)*, to define exploration strategies for search and rescue. Using decision-theoretical tools, on the one hand, contributes to the further assessment of the science of robotics and, on the other hand, provides practical advantages in the definition of effective exploration strategies. Although MCDM has been already applied to map building with a single robot [5], we deem that its application to multirobot search and rescue represents a significant contribution since it addresses a more challenging setting for exploration strategies, where the primary objective is not to build an accurate map of the physical space but to search the environment for locating the largest number of victims in a limited amount of time. Differently from map building, in search and rescue settings operations must be performed quickly, privileging the amount of explored area over the map quality. To the best of our knowledge, this is the first attempt to apply MCDM to search and rescue.

We consider a situation in which a team of robots have to search an initially unknown environment for victims. Since no *a priori* knowledge about the possible locations of the victims is assumed to be available, we can reduce the problem of maximizing the number of victims found in a given time interval to the problem of maximizing the amount of area covered by robots' sensors in the same time interval. Broadly speaking, the robots operate according to the following steps: (a) they perceive the surrounding environment, (b) they integrate the perceived data within a map representing the environment known so far, (c) they decide where to go next, and (d) they go there and start again from (a). We propose to use MCDM for addressing step (c), namely for defining the exploration strategy. In our experiments, we implemented the proposed approach as a modification of a publicly available controller used for the RoboCup Rescue Virtual Robots Competition [18]. In this way, on the one hand, we can focus on the development of exploration strategies (step (c)) exploiting an already tested framework for steps (a), (b), and (d) and, on the other hand, we can fairly compare our strategies with that originally used in [18].

2. RELATED WORK

Robotic exploration is a broad concept that can be defined as a process that discovers unknown features in environments by means of mobile robots. Exploration is employed in several tasks, like map building [16], search and rescue [15], and coverage [8]. For example, in map building the features to be discovered are the ob-

stacles and the free space, while in search and rescue can be the locations of victims or fires. *Exploration strategies* are used to move autonomous robots around environments in order to discover their features. In this paper, we are interested in exploration strategies employed for discovering the physical structure of environments that are initially unknown. In these scenarios, we do not know *ex ante* the complete set of the possible locations that the robots can reach. We explicitly note that, as a consequence, we cannot employ some exploration strategies, like those proposed in [11] and [13], which require an *a priori* knowledge on the possible observation locations. In the following, we survey a representative sample of the countless exploration strategies that have been proposed in literature.

Unsurprisingly, most of the work on exploration strategies for discovering the physical structure of environments has been done for map building. The mainstream approach models exploration as an incremental Next Best View (NBV) process, i.e., a repeated greedy selection of the next best observation location. Usually, at each step, an NBV system considers a number of candidate locations on the frontier between the known free space and the unexplored part of the environment (in such a way they are reachable from the current position of the robot) and selects the best one [20]. The most important feature of an exploration strategy is how it evaluates candidate locations in order to select the best one.

In evaluating candidate locations, different criteria can be used. A simple one is the distance from the current position of the robot [20], according to which the best observation location is the nearest one. However, most works combine different criteria in more complex utility functions. For example, in [14] the cost of reaching a candidate location p is linearly combined with its benefit. Measuring the cost as the distance $d(p)$ of p from the current location of the robot and the benefit as an estimate of the new information $A(p)$ acquirable from p , the global utility of p is computed as:

$$u(p) = A(p) - \beta d(p), \quad (1)$$

where β balances the relative weight of benefit versus cost (authors show that choosing β within the interval [0.01, 50] does not cause significant variations in the exploration performance). Another example of combination of different criteria is [9], in which distance $d(p)$ and the expected information gain $A(p)$ of a candidate location p are combined in an exponential function

$$u(p) = A(p)e^{-\lambda d(p)} \quad (2)$$

(where λ is a parameter that weights the two criteria). In [1], a technique based on relative entropy is used to combine traveling cost and expected information gain. In [17], several criteria (such as uncertainty in landmark recognition and number of visible features) are combined in a multiplicative function. In [12], traveling cost to reach a location is used as the main criterion for evaluating candidate locations, while the utility of the locations (calculated according to the proximity of other robots) is used as a tie-breaker.

The above strategies aggregate different criteria in utility functions that are defined *ad hoc* and are strongly dependent on the criteria they combine. In [2], the authors dealt with this problem and proposed a more theoretically-grounded approach based on multi-objective optimization, in which the best candidate location is selected on the Pareto frontier. Besides distance and expected information gain, also overlap is taken into account. This criterion is related to the amount of old information that will be acquired again from a candidate location. Maximizing the overlap can improve the self-localization of the robot. The work presented in this paper follows the same theoretically-grounded approach and, as described

in Sections 3 and 4, tries to employ MCDM in search and rescue applications.

Compared with exploration strategies for map building, only few works proposed exploration strategies for autonomous search and rescue. A work that explicitly addressed this problem is [18], which proposes to combine the distance $d(p)$, the expected information gain $A(p)$, and the probability of a successful communication $P(p)$ from a candidate location p in the following utility function:

$$u(p) = \frac{A(p)P(p)}{d(p)}. \quad (3)$$

This strategy has been employed, with good results, in different RoboCup Rescue Virtual Robots Competitions. In this work we experimentally compare the exploration strategies developed with our approach with that proposed in [18], which is explicitly devoted to the same goal. Another exploration strategy for search and rescue is reported in [6], where a formalism based on Petri nets is used to exploit *a priori* information about the victims' distribution (e.g., if they are uniformly spread or concentrated in few clusters) to improve the search.

3. MULTI-CRITERIA DECISION MAKING

When designing an effective exploration strategy for exploring initially unknown environments, the main challenge is to achieve a good global (long-term) performance by means of local (short-term) decisions that are made on the basis of partial knowledge. In our scenario, the partial knowledge is given by the current map built by the robots and short-term decisions are made by evaluating a number of alternatives, i.e., candidate observation locations on the frontiers between the explored and unexplored space, and by selecting the best one. The "goodness" of an observation location can be measured with respect to multiple criteria, as we have seen in the previous section. The number of criteria that can be considered is, in principle, unlimited. As the tasks the robots perform become more complex (think, for example, of an exploring robot that has also to find victims, localize fire sources, communicate with a base station, and so on), this number is likely to increase.

In this work, we explicitly consider the evaluation of candidate locations as a multi-objective (or multi-criteria) optimization problem. We have a set C of candidate locations among which we want to choose the "best" one. We denote the set of n criteria considered in the evaluation process as $N = \{1, 2, \dots, n\}$. Given a candidate $p \in C$ we denote with $u_i(p) \in I$ its *utility* with respect to criterion $i \in N$, where $I \subseteq \mathbb{R}$ represents the set of possible utility values. Note that we assume that all utilities have values over the same set I . The larger the utility $u_i(p)$, the better location p satisfies criterion i . Each candidate p can be associated to a vector of n elements, namely its utilities, $u_p = (u_1(p), u_2(p), \dots, u_n(p))$. The problem of selecting the "best" candidate observation location comes down to the problem of selecting the optimal candidate location p^* from C .

Dealing with this multi-criteria scenario, the optimality of candidates involves the concept of *Pareto frontier*. Formally, the Pareto frontier of C can be defined as the largest subset $P \subseteq C$ such that for every $p \in P$ there is not any candidate $q \in C$ with $u_i(q) > u_i(p)$ for all $i \in N$. A candidate $q \in C \setminus P$ is said to be *Pareto-dominated* and can be safely discarded, since at least a preferable candidate is guaranteed to exist in P . Therefore, choosing a candidate on the Pareto frontier P is a fundamental requirement to select a "good" candidate. The actual selection is performed via a *global utility function* $u(p) = f(u_p) = f(u_1(p), u_2(p), \dots, u_n(p))$ that combines together utilities in an aggregate value (well-known ex-

amples are the arithmetic and weighted mean). Since computing the Pareto frontier P can be computationally expensive (especially when the number of candidates grows), the selection is usually done by looking directly at the initial set C , namely $p^* = \arg \max_{p \in C} f(u_p)$. It can be easily shown that if $f()$ is a non-decreasing function in every one of its n arguments, then p^* is guaranteed to be on the Pareto frontier. As the previous section shows, the mainstream approach followed in literature to define global utility functions is to combine a pre-determined number of criteria in an *ad hoc* form. Despite it is not explicitly mentioned, almost all these methods are Pareto optimal, since a non-decreasing global utility function is a “natural” choice.

In the following section, we describe Multi-Criteria Decision Making (MCDM) as a general method for defining global utility functions and we discuss some of its advantages and properties that make it a valid tool for defining exploration strategies for autonomous mobile robots.

3.1 Combining Criteria with the Choquet Integral

We introduce and motivate the proposal of MCDM by considering the important aspect of the dependency between criteria, that is often neglected by global utility functions. Criteria that are used to evaluate candidate locations are not always independent. For example, think of criteria that estimate the same feature using different methods, like two criteria that estimate the distance of a candidate location from the current position of the robot according to the Euclidean and Manhattan distance. Intuitively, when combining them into a global utility function, their overall contribution to the global utility of a candidate location should be less than the sum of their individual ones. In this case, a *redundancy* relation holds between criteria. A dual situation occurs when two or more criteria are very different and, in general, can be hardly optimized together. In this case, a *synergy* relation holds between criteria, and their overall contribution should be considered larger than the sum of the individual ones. An example involves the estimated information gain and the overlap. These criteria can be considered synergic, since large utilities for both are very difficult to achieve by a single candidate and candidates that satisfy both criteria reasonably well should be preferred to candidates that satisfy them in an unbalanced way. In order to consider these issues we need a way to define a global utility function that accounts for redundancy and synergy between criteria when combining them. MCDM provides a general aggregation method which can deal with this and with other aspects and that exploits the *Choquet integral* to compute global utilities [10]. Let us introduce it.

We first introduce a (total) function $\mu : \mathcal{P}(N) \rightarrow [0, 1]$ ($\mathcal{P}(N)$ is the power set of set N) with the following properties: $\mu(\{\emptyset\}) = 0$, $\mu(N) = 1$, and if $A \subset B \subset N$, then $\mu(A) \leq \mu(B)$. That is, μ is a normalized *fuzzy measure* on the set of criteria N that will be used to associate a weight to each group of criteria. The weights specified by the definition of μ describe the dependency relations that hold for each group of criteria. Criteria belonging to a group $G \subseteq N$ are said to be redundant if $\mu(G) < \sum_{i \in G} \mu(i)$, synergic if $\mu(G) > \sum_{i \in G} \mu(i)$, and independent otherwise.

The global utility $f(u_p)$ for a candidate p is computed as the discrete Choquet integral $\mathcal{C}()$ with respect to the fuzzy measure μ using p 's utilities:

$$f(u_p) = \mathcal{C}(u_p) = \sum_{j=1}^n (u_{(j)}(p) - u_{(j-1)}(p)) \mu(A_{(j)}), \quad (4)$$

where $(j) \in N$ indicates the j -th criterion according to an increas-

ing ordering with respect to utilities, i.e., after that criteria have been permuted to have, for candidate p ,

$$u_{(1)}(p) \leq \dots \leq u_{(n)}(p) \leq 1.$$

It is assumed that $u_{(0)}(p) = 0$. Finally, the set $A_{(j)}$ is defined as

$$A_{(j)} = \{i \in N | u_{(j)}(p) \leq u_i(p) \leq u_{(n)}(p)\}.$$

Using $\mathcal{C}(u_p)$ to compute global utilities allows to consider criteria's importance and their mutual dependency relations.

3.2 Some Properties of MCDM

In this section, we discuss a number of properties of the proposed MCDM approach. A first general feature of the Choquet integral is that, differently from *ad hoc* global utility functions, it can be applied to any number of criteria. Indeed, rigorously speaking, $\mathcal{C}()$ as defined in (4) is not an aggregation function, for which the number of arguments is fixed *a priori*, but an *aggregation operator*. An aggregation operator is a collection of aggregation functions, one for each number n of criteria to be combined. For example, the arithmetic and weighted means are aggregation operators since they basically specify an aggregation technique for every possible number of criteria, while global utility functions like (2) and (3) are aggregation functions suitable only for the set of criteria they have been tailored for. In this sense, we can say that an aggregation operator is more general than an aggregation function. An obvious advantage of using an aggregation operator instead of an aggregation function is the increased flexibility, because adding and removing criteria can be accomplished preserving the way in which they are combined. As we will discuss in the next sections, this feature enables easy refinements of the exploration strategies and facilitates some experimental activities such as assessing the impact of removing or including a criterion.

$\mathcal{C}(u_p)$ enjoys several other properties [10]. Here, we briefly discuss some properties that are significant in connection with the definition of exploration strategies and that characterize MCDM as a suitable approach to define global utility functions.

Increasing monotonicity in each argument

For all $u_p, u'_p \in I^n$,

- if $\forall i \in N, u_i(p) \leq u'_i(p)$, then $\mathcal{C}(u_p) \leq \mathcal{C}(u'_p)$,
- if $\forall i \in N, u_i(p) < u'_i(p)$, then $\mathcal{C}(u_p) < \mathcal{C}(u'_p)$.

This property can be exploited to guarantee that the maximization of $\mathcal{C}()$ over the set of candidate locations C will select a Pareto optimal candidate. As we discussed before, almost all aggregation functions proposed in literature for exploration strategies satisfy this property.

Stability for linear transformations

For all $u_p \in I^n$ and $r, s \in \mathbb{R}$ with $r > 0$ such that, for all $i \in N$, $ru_i(p) + s \in I$, it holds that

$$\begin{aligned} \mathcal{C}(ru_1(p) + s, ru_2(p) + s, \dots, ru_n(p) + s) = \\ r\mathcal{C}(u_1(p), u_2(p), \dots, u_n(p)) + s. \end{aligned}$$

This property ensures the independence of the particular scale in which utilities are measured (up to a linear transformation). In this paper we assume, without any loss of generality, that utilities have values in $I = [0, 1]$; however, any other common scale would have been equivalent. In general, this property is rarely satisfied by aggregation functions proposed in literature, where often criteria are measured with respect to different scales and combined without any normalization (see, for example, [9] and [18]).

Continuity

Given n , the corresponding aggregation function $\mathcal{C}(\cdot)$ is continuous on I^n . This property prevents the global utility to exhibit irregular variations with respect to small changes of the utility values that are aggregated. When the global utility is computed by adopting exponential or fractional functions (see (2) and (3)), this property is satisfied.

Idempotence

If, for a given p , all $u_i(p) = u \in I$, then

$$\mathcal{C}(u_1(p), u_2(p), \dots, u_n(p)) = \mathcal{C}(u, u, \dots, u) = u.$$

This property assures a sort of consistency, namely, if all the criteria are satisfied with the same degree u , then the global utility is u . This property is rarely exhibited by the aggregation functions used in literature, with the drawback that the particular form in which criteria are combined can introduce a bias in the evaluation, for example by implicitly giving more importance to some criteria to the detriment of others.

3.3 Generality of MCDM

Another important advantage of MCDM is its generality. Indeed, different aggregation operators turn out to be particular cases of the Choquet integral, up to a proper choice of weights for the fuzzy measure μ . For instance, a class of aggregation operators that can be expressed with the Choquet integral are *weighted means*. A weighted mean is defined as $\sum_{i=1}^n w_i u_i(p)$ where w_i is the weight of criterion i and $\sum_{i=1}^n w_i = 1$. This aggregation operator can be obtained from Choquet integral by setting $\mu(\{i\}) = w_i$ for all $i \in N$ and by constraining μ to be additive:

$$\mu(S) = \sum_{i \in S} w_i \quad \forall S \in \mathcal{P}(N).$$

Note that additivity of μ reflects independence between criteria, namely joint contributions are exactly the sum of marginal ones. Therefore, weighted means should be considered suitable when such independence between criteria holds. Moreover, the arithmetic mean and the k -th criterion projection can be obtained as further particular cases of weighted means by imposing $w_i = 1/n \forall i \in N$ and $w_k = 1, w_i = 0 \forall i \in N \setminus \{k\}$, respectively. In the context of exploration, this means that the strategy proposed in [14] and based on (1) can be viewed as a special case of MCDM-based exploration strategies. Moreover, also the global utility function proposed in [12] can be viewed as a special case of MCDM, basically being a k -th criterion projection.

A second class of aggregation operators that are special cases of the Choquet integral is composed of *ordered weighted means*. An ordered weighted mean is defined as $\sum_{j=1}^n w_j u_{(j)}(p)$ (i.e., a weighted mean in which w_j is the weight of the j -th criterion according to an increasing ordering of utilities). An ordered weighted mean aggregation operator can be obtained from the Choquet integral by setting $\mu(\{i\}) = w_i$ for all $i \in N$ and by defining $\mu(S)$ according to:

$$\mu(S) = \sum_{i=n-|S|+1}^n w_i \quad \forall S \in \mathcal{P}(N).$$

Some further particular cases of ordered weighted means that can be modeled with a proper choice of weights w_i are the minimum and maximum (when $w_1 = 1$ and $w_n = 1$, respectively), the median (when $w_{\frac{n}{2}} = w_{\frac{n}{2}+1} = 0.5$ and n is even or when $w_{\frac{n+1}{2}} = 1$ and n is odd), and the arithmetic mean excluding the two extremes

(when $w_1 = w_n = 0$ and $w_i = \frac{1}{n-2} \forall i \in N \setminus \{1, n\}$). This shows the possibility offered by MCDM of obtaining completely different global utility functions (and, as a consequence, different behaviors of the robot) by simply setting weights μ . In this sense, we say that MCDM constitutes a general approach for defining exploration strategies.

4. MCDM-BASED EXPLORATION STRATEGIES FOR SEARCH AND RESCUE

We apply the proposed MCDM approach to search and rescue, where mobile robots are deployed in an initially unknown environment with the goal to explore it and locate human victims within a limited amount of time. As discussed in Section 1, this application domain offers a challenging scenario to test exploration strategies.

We implemented MCDM-based exploration strategies in an existing robot controller for search and rescue applications. We looked at the participants to the RoboCup Rescue Virtual Robots Competition where different teams compete in developing simulated robotic platforms operating in Urban Search And Rescue scenarios simulated in USARSim [7] (an high fidelity 3D robot simulator). From an analysis based on availability of code and performance obtained in the competition, we selected the controller developed by the Amsterdam and Oxford Universities (Amsterdam Oxford Joint Rescue Forces, AOJRF¹) for the 2009 competition [19]. The reasons for implementing MCDM-based exploration strategies in an existing controller are that we can focus only on the exploration strategies, exploiting existing and tested methods for navigation, localization, and mapping and that we have a fair way to compare our exploration strategies with that originally used in the controller. In the following we describe the original controller and how we modified it to implement MCDM-based strategies.

4.1 The AOJRF Controller

In this section, we describe some of the controller's features that are relevant to the scope of this paper (please refer to [18] for a complete description).

The controller manages a team of robots. The robotic platform used is a Pioneer 3AT, whose basic model and sensors are provided with the USARSim simulator. The map of the environment is maintained by a base station, whose position is fixed in the environment, and to which robots periodically send data. The map is two-dimensional and represented by two occupancy grids. The first one is obtained with a small-range (typically 3 meters) scanner and constitutes the *safe area*, i.e., the area where the robots can safely move. The second one is obtained from maximum-range scans (typically 20 meters) and constitutes the *free area*, i.e., the area which is believed to be free but not yet safe. Moreover, a representation of the *clear area* is also maintained as a subset of the safe area that has been checked for the presence of victims (this task is accomplished with simulated sensors for victim detection). Given a map represented as above, a set of boundaries between safe and free regions are extracted and considered as frontiers. For each frontier, the middle point is considered as a candidate location to reach. The utility of a candidate location p is evaluated by combining the following criteria:

- $A(p)$ is the amount of the free area beyond the frontier of p computed according to the free area occupancy grid;
- $P(p)$ is the probability that the robot, once reached p , will be able to transmit information (such as the perceived data or

¹<http://www.jointrescueforces.eu/>

the locations of victims) to the base station (whose position in the environment is known), this criterion depends on the distance between p and the base station;

- $d(p, r)$ is the distance between p and current position of robot r , this criterion can be calculated with two different methods: $d_{EU}()$, using the Euclidean distance, and $d_{PP}()$, using the exact value of the distance returned by a path planner.

Given these criteria, the global utility for a candidate p is calculated using function (3). We will refer to the exploration strategy using this global utility function as the “AOJRF strategy”.

The allocation of candidate locations to robots is performed with the following algorithm, which is executed by each robot independently, knowing (from the base station) the current map and the positions of other robots [18]:

1. compute the global utility $u(p, r)$ of allocating each candidate p to each robot r using (3) where $d(p, r)$ is calculated using the Euclidean distance $d_{EU}()$ (namely using an underestimate of the real distance),
2. find the pair (p^*, r^*) such that the previously computed utility is maximum, $(p^*, r^*) = \arg \max_{p, r} u(p, r)$,
3. re-compute the distance between p^* and r^* using $d_{PP}()$ with the path planner (namely considering the real distance) and update the utility of (p^*, r^*) using such exact value instead of the Euclidean distance,
4. if (p^*, r^*) is still the best allocation, then allocate robot r^* to location p^* , otherwise go to Step 2,
5. eliminate robot r^* and candidate p^* and go to Step 2.

The reason behind the utility update of Step 3 is that computing $d_{PP}()$ requires a considerable amount of time. Doing this for all the candidate locations and all robots would be not affordable in the rescue competition, since a maximum exploration time of 20 minutes is enforced.

4.2 Developing MCDM-based strategies

We now describe the changes we made to the original controller to include our MCDM-based strategies.

MCDM	criteria	$\mu()$	criteria	μ
	A	0.5	A, d	0.95
d	0.3	A, P	0.7	
P	0.2	d, P	0.4	

MCDMb	criteria	$\mu()$	criteria	$\mu()$
	A	0.4	d, P	0.25
d	0.25	d, b	0.35	
P	0.1	P, b	0.25	
b	0.25	A, d, P	0.75	
A, d	0.75	A, d, b	0.9	
A, P	0.5	A, P, b	0.75	
A, b	0.65	d, P, b	0.45	

MCDMw	criteria	$\mu_1()$	$\mu_2()$
	A	0.6	0.4
d	0.1	0.5	
P	0.3	0.1	
A, d	0.8	0.95	
A, P	0.9	0.5	
d, P	0.3	0.5	

Table 1: Weights used for the MCDM-based strategies.

The first MCDM-based strategy we propose adopts the same criteria of the AOJRF strategy (i.e., A , P , and d , as described above), but combines them with the MCDM approach. Basically, we replace function (3) with function (4), with the weights reported in Tab. 1 (top-left). We call this the “MCDM strategy”.

Choosing a particular set of weights can be tricky. In this phase, the designer considers the application domain and defines the importance of single and groups of criteria. We remark that searching

for the “best” set of weights is an ill-posed problem in the context of MCDM. MCDM is not a method to determine the best exploration strategy, but provides a flexible and general tool to combine criteria. Therefore, we assigned weights manually, considering the search and rescue context. For example, the MCDM strategy assigns more importance to A than to P and d (see Tab. 1 (top-left)), pushing the robot to discover new areas, even covering long distances or risking a loss of communication. The joint contribution of d and P is inhibited by establishing redundancy between them. On the other side, a synergy holds between d and A , privileging locations satisfying these criteria in a balanced way. This manual method for assigning weights does not scale well with the number n of criteria. Indeed, $2^n - 2$ weights have to be assigned. However, specification of weights is done at design-time and there are semi-automated techniques to compute weights for large sets of criteria [10].

To apply MCDM, utilities have to be normalized to the chosen common scale $I = [0, 1]$. We note that the robot’s decision at any step depends only on C and not on previous decisions and previous sets of candidate locations. Hence, we use a linear relative normalization. For example, given a robot r , the utility of a candidate p related to the distance $d()$ is normalized using $u_d(p, r) = 1 - (d(p, r) - \min_{q \in C} d(q, r)) / (\max_{q \in C} d(q, r) - \min_{q \in C} d(q, r))$. This poses a problem for normalizing the updated utility in Step 3, since it would require to determine the path for every candidate location, making the 20 minutes limit too strict to achieve an acceptable performance (recall that $d_{PP}()$ is computationally expensive). To deal with this problem we use the following procedure in Step 3: once computed $d_{PP}(p^*, r^*)$, we normalize it by using the previously calculated values $d_{EU}(p, r^*)$ for other candidates $p \in C$.

The second MCDM-based strategy we propose shows the flexibility of MCDM in adding a new criterion, i.e., the robot’s battery remaining charge b . Explicitly considering the battery can improve exploration by preventing the robot from making decisions it cannot complete (e.g., selecting a location not reachable with the residual energy). To compute $u_b(p)$ we need an estimate of the energy spent for reaching p . We consider a very simple model in which the power consumption is translated in a time interval. In order to estimate the time needed to reach a location p we consider the path the robot should follow in terms of linear segments and rotations. By approximating the linear and angular velocities of the robot as constants, we can derive estimates of the time $b(p)$ needed to reach p . Obviously, the smaller $b(p)$ the larger $u_b(p)$. Notice that b and d show an evident dependency relation given by the fact that long traveling distances often correspond to long times. However, despite this similarity, including b in the set of criteria can, to some extent, provide more informed decisions since it captures also the difficulty for covering a path which generally is not captured by d (consider, for example, short but winding paths that could require lot of time and battery). Modeling a redundancy relation between these two criteria is the proper way to include both of them in the decision-making process without unbalancing decisions toward the common selection principle encoded in b and d . We denote the strategy including b as “MCDMb strategy”, whose weights are reported in Tab. 1 (top-right). As it can be seen, the weight assigned to the set $\{d, b\}$ is lower than the sum of weights of b and d . Redundancy and synergy are also defined on sets of more than two criteria; for example, criteria d , P , and b are redundant and the weight of the set $\{d, P, b\}$ is smaller than the sum of the weights of its elements.

We also show how MCDM can be adopted for defining different behaviors in exploration. Broadly speaking, a behavior defines the

preferences according to which the robot selects observation locations. Given a set of criteria, a behavior is associated to the particular set of weights of those criteria. By changing the weights during exploration, we can switch between different behaviors, varying the criteria’s importance that drive robot decisions. This technique allow us to improve the exploration strategy’s adaptability to different situations. Hence, we define a third MCDM-based strategy, called the “MCDMw strategy”, whose weights are reported in Tab. 1 (bottom). This strategy encloses two different behaviors, given by the sets of weights denoted as μ_1 and μ_2 , defined over the original set of criteria of the MCDM strategy (i.e., A , P , and d , as described above). In addition, we define the following policy for switching through behaviors. The weights defined by $\mu_1()$ are used during the first 10 minutes of search while those defined by $\mu_2()$ are used during the last 10 minutes. The first set of weights encodes an aggressive behavior oriented towards the maximization of the new area. This behavior is reasonable during the first part of the search when a long remaining time is left and the robot can privilege the amount of new area even if long paths have to be followed. Differently, the second set of weights induces a more conservative behavior. This behavior accounts for the fact that remaining time is short and gives more importance to distance ($\mu_1(d) = 0.1$ while $\mu_2(d) = 0.5$).

5. EXPERIMENTAL EVALUATION

In the first experiments we evaluate the performance of the MCDM strategy when compared with other strategies. We consider the AOJRF strategy (corresponding to (3)), the WS strategy (corresponding to (1) with $\beta = 1$), and the DIST strategy, by which locations are selected simply by minimizing d (i.e., choosing always the nearest location). AOJRF and WS are continuous and increasingly monotonic aggregation functions. These two strategies guarantee a Pareto optimal selection, however AOJRF strategy lacks in flexibility since including further criteria would require to re-define the aggregation technique, while WS can be considered as a special case of MCDM-based strategies (see Section 3.3). Nevertheless, AOJRF and WS have been proved to achieve good results in practice, therefore, by comparing the MCDM strategy with them, we aim at deriving insights on how performance changes when using a more theoretically-grounded way to define global utility functions. DIST is a very simple strategy that can be viewed as a particular case of MCDM-based strategy. Indeed, it can be obtained by restricting the set of criteria to the singleton d (see Section 3.3). By comparing MCDM and DIST we aim at confirming that making more informed local decisions actually results in a better global performance.

We considered teams of one or two robots, as in [18] (note that the maximum number of robots allowed in the RoboCup Rescue Virtual Robots Competition is 4). The robots are deployed in the two indoor environments of Fig. 1 that show different characteristics. Map A is cluttered and composed of corridors and many rooms, while Map B is characterized by the presence of open spaces. A configuration is defined as an environment, a team of robots deployed in it, and the exploration strategy adopted. For each configuration, we executed 10 runs (with randomly selected starting locations for the robots) of 20 minutes each. We assess performance by measuring the amount of free, safe, and clear area at each minute of the exploration. Due to space limitations, we report only data on safe area (free area is less significant and clear area is similar to the safe area).

Figs. 2 and 3 show the results of the first experiments with a team of one and two robots, respectively. Histograms compare the number of runs and two strategies obtained the largest amount of safe area at the end of the 20 minutes exploration. Graphs show how the

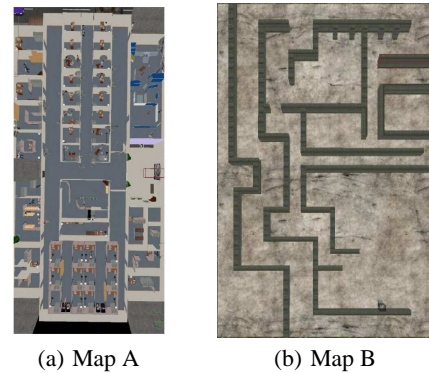


Figure 1: The maps used for tests.

mapped safe area varies with time (each point is the average over 10 runs).

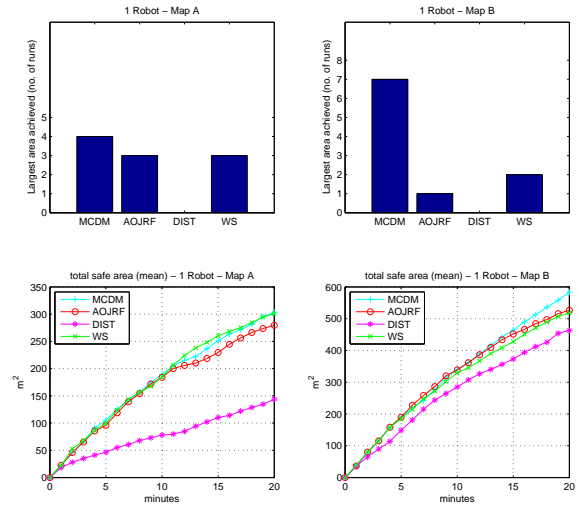


Figure 2: Comparison between MCDM and other exploration strategies with one robot.

The MCDM strategy discovered the largest area in the majority of runs, outperforming (on average) other strategies. According to an ANOVA test, the averages of the total safe area (in Map A) are statistically significantly different between DIST and each one of the other three strategies. Differences between MCDM, AOJRF, and WS are not statistically significant in Map A. In Map B, the MCDM strategy shows a statistically significant difference when compared to DIST and AOJRF, while the statistical difference between MCDM and WS is slightly acceptable. These findings reflect an interesting insight associated to the different characteristics of the two environments. Map A is cluttered and, exploring it, the robots deal with a relatively large number of frontiers among which to choose (30 candidate locations on average at each step with one robot and 40 with two robots). Map B is characterized by open spaces, resulting in a smaller number of candidate frontiers (5 candidate locations on average at each step with one robot and 8 with two). However, despite their large number, frontiers in Map A are very similar in the contribution they can give to the

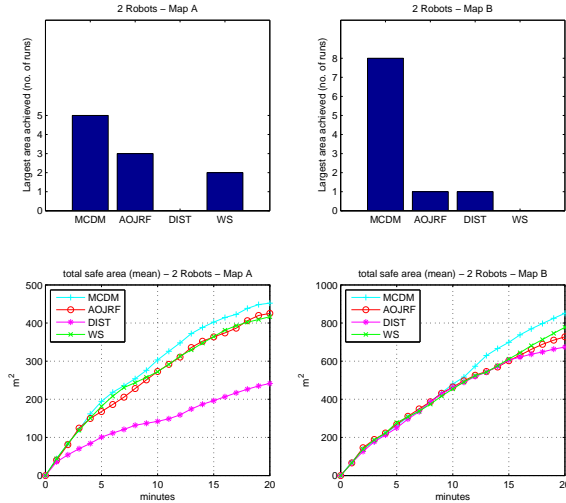


Figure 3: Comparison between MCDM and other exploration strategies with two robots.

explored area. Differently, in Map B the situation in which one alternative is remarkably better than others is more frequent. Consider, for instance, a frontier that lies close to an obstacle (from where an observation will return a small new area) and another one in front of an open space. In such situation, the benefits provided by a “right choice” would be more evident. This is what happens during the exploration of Map B, showing why differences between strategies are statistically significant in this environment. This basically confirms the single robot results presented in [5], enforcing the idea that when very different alternatives are present and making a good choice is very rewarding, MCDM-based exploration strategies achieve satisfactory results.

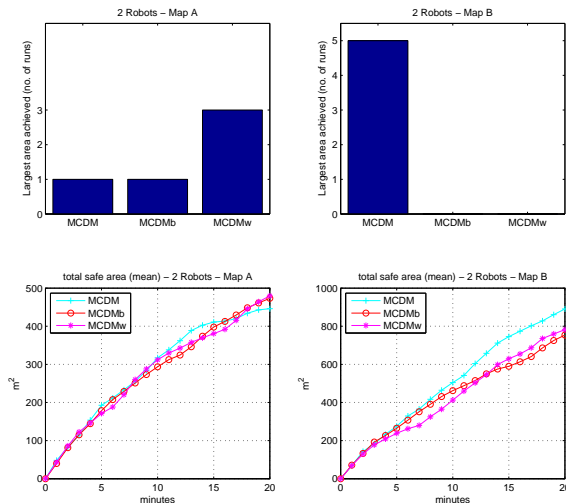


Figure 4: Comparison between the MCDM-based strategies.

Fig. 4 shows the performance of the three MCDM-based exploration strategies with two robots (we omit results with one robot, for which the same considerations can be drawn). A first comparison that is worth doing is between MCDM and MCDMb, to assess the impact of b 's inclusion in evaluating a location. When adopted for exploring Map A, these two strategies performed similarly, not showing any statistically significant difference in the total safe area. However, the effects of introducing criterion b can be noted by looking at the final maps built by the robots. A representative example is shown in Fig. 5, which reports the two maps obtained with MCDM and MCDMb after a run. Considering that the criterion b pushes the robots to discard locations that require complicate paths with several rotating maneuvers, the robots save time avoiding to deeply explore corners, rooms, and other cluttered parts of the environment, preferring corridors and open spaces. The result is that the obtained map, from the one hand, is less precise but, from the other hand, is more representative of the general topology of the environment. This kind of map can be more useful to first responders in giving a broad idea of the topology of the environment (as discussed in [3]). The introduction of the criterion b does not show the same qualitative behavior in Map B, where the presence of open spaces makes intricate paths very rare. The employment of this criterion in an open space is not justified by the characteristics of the environment, showing an example where “too-much informed” local decisions could achieve a not so good global performance.

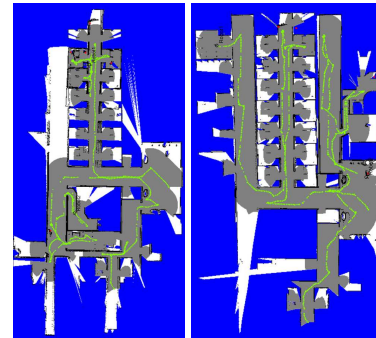


Figure 5: An example of maps obtained after an exploration.

Adopting different behaviors with the MCDMw strategy led to the best results in Map A. Roughly speaking, this strategy combines the benefits of MCDM and MCDMb strategies. In the first half of the exploration a more aggressive behavior is adopted, trying to maximize the explored area. Then, as the residual time decreases, the strategy becomes more conservative, trying to save time avoiding cluttered zones. In Map B, the employment of μ_1 in the first part of the exploration showed the main drawback of a very aggressive behavior: its vulnerability to decisions that happen to be not as good as expected. In a number of situations, μ_1 pushed the robots to cover long distances for reaching locations with potentially large amounts of new area that, due to information gain estimation errors, were not so informative once reached. This is the reason why MCDM and MCDMw curves are relatively separated in the first 10 minutes of exploration (Fig. 4).

Fig. 6 depicts an example of paths followed by a robot when employing the three MCDM-based strategies. The starting location in all the three cases is at the center of the top corridor. All the strategies initially drive the robot toward the right part of the top corridor until the first difference can be observed in the path of MCDMw. Its

aggressive behavior pushed the robot to go back at the intersection with the vertical corridor to obtain a wide view over the free space. MCDM and MCDMb start to significantly differ in the bottom end of the central vertical corridor. More precisely, MCDMb's path resulted more regular than that of MCDM. Indeed, MCDM drove the robot to explore a sequence of rooms while, with MCDMb, the robot chose to enter the bottom horizontal corridor. MCDMw's paths avoided all the rooms in the right part of the environment (first 10 minutes) but performed a more detailed exploration in the left part of the map (last 10 minutes of the exploration). This example shows how obtained paths are coherent with the design principles of each strategy and demonstrates that the decision-theoretic framework of the MCDM-based strategies can provide some level of predictability.



Figure 6: Example of paths of MCDM-based strategies.

From our results, we can say that MCDM can be an effective method for defining good exploration strategies in search and rescue applications. Local decisions made with MCDM-based exploration strategies resulted in a comparable and sometimes better performance, when compared to other exploration strategies proposed in literature. In particular, MCDM showed significant improvements in situations (like those faced in Map B) where making the right decision is more rewarding. In addition, MCDM presents a remarkable flexibility in composing criteria that can be exploited to add new criteria or to define multi-behavioral strategies that can adapt to different situations.

6. CONCLUSIONS

In this paper, we have presented the application of the MCDM decision-theoretic approach to the definition of exploration strategies for search and rescue. We have shown that MCDM provides a general and flexible way for developing utility functions for evaluating candidate observation locations. Experimental results show that MCDM-based exploration strategies achieve a good performance, when compared with *ad hoc* strategies used in exploration.

Possible future work includes the development of automatic techniques to set the values of weights in MCDM, in order to further simplify the inclusion of new criteria in the evaluation of candidate locations, and the application of MCDM-based strategies to other domains, like planetary exploration. Another interesting direction is working on the robot-frontier allocation, trying to achieve a closer integration between evaluation of candidate locations and coordination of robots.

7. REFERENCES

[1] F. Amigoni and V. Caglioti. An information-based

exploration strategy for environment mapping with mobile robots. *ROBOT AUTON SYST*, 5(58):684–699, 2010.

[2] F. Amigoni and A. Gallo. A multi-objective exploration strategy for mobile robots. In *Proc. ICRA*, pages 3861–3866, 2005.

[3] B. Balaguer, S. Balakirsky, S. Carpin, and A. Visser. Evaluating maps produced by urban search and rescue robots: Lessons learned from robocup. *AUTON ROBOT*, 27(4):449–464, 2009.

[4] S. Balakirsky, C. Scrapper, S. Carpin, and M. Lewis. Usarsim: a robocup virtual urban search and rescue competition. In *Proc. of SPIE*, 2007.

[5] N. Basilico and F. Amigoni. Exploration strategies based on multi-criteria decision making for an autonomous mobile robot. In *Proceedings of the European Conference on Mobile Robots (ECMR)*, pages 259–264, 2009.

[6] D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi. Multi-objective exploration and search for autonomous rescue robots. *J FIELD ROBOT*, 24(8-9):763–777, 2007.

[7] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. Usarsim: a robot simulator for research and education. In *Proc. ICRA*, pages 1400–1405, 2007.

[8] H. Choset for robotics: A survey of recent results. *Ann. Math. Artif. Intell.*, 31(1-4):113–126, 2001.

[9] H. González-Baños and J.-C. Latombe. Navigation strategies for exploring indoor environments. *INT J ROBOT RES*, 21(10-11):829–848, 2002.

[10] M. Grabisch and C. Labreuche. A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid. *4OR-Q J OPER RES*, 6(1):1–44, 2008.

[11] K. H. Low, J. Dolan, and P. Khosla. Adaptive multi-robot wide-area exploration and mapping. In *Proc. AAMAS*, pages 23 – 30, 2008.

[12] A. Marjovi, J. Nunes, L. Marques, and A. de Almeida. Multi-robot exploration and fire searching. In *Proc. IROS*, pages 1929 –1934, 2009.

[13] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser. Efficient informative sensing using multiple robots. *J ARTIF INTELL RES*, 34(1):707–755, 2009.

[14] C. Stachniss and W. Burgard. Exploring unknown environments with mobile robots using coverage maps. In *Proc. IJCAI*, pages 1127–1134, 2003.

[15] S. Tadokoro. *Rescue Robotics*. Springer-Verlag, 2010.

[16] S. Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*, pages 1–35. 2002.

[17] B. Tovar, L. Munoz-Gomez, R. Murrieta-Cid, M. Alencastre-Miranda, R. Monroy, and S. Hutchinson. Planning exploration strategies for simultaneous localization and mapping. *ROBOT AUTON SYST*, 54(4):314 – 331, 2006.

[18] A. Visser and B. A. Slamet. Including communication success in the estimation of information gain for multi-robot exploration. In *Proc. WiOPT*, pages 680–687, 2008.

[19] A. Visser *et al.* Amsterdam Oxford joint rescue forces - team description paper - Virtual Robot competition - Rescue simulation league - robocup 2009. In *Proc. of RoboCup*, 2009.

[20] B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proc. CIRA*, pages 146–151, 1997.

Simulation-based Temporal Projection of Everyday Robot Object Manipulation

Lars Kunze, Mihai Emanuel Dolha, Emitza Guzman, Michael Beetz
Intelligent Autonomous Systems Group
Technische Universität München
80333 Munich, Germany
{kunzel,dolha,ortega,beetz}@cs.tum.edu

ABSTRACT

Performing everyday manipulation tasks successfully depends on the ability of autonomous robots to appropriately account for the physical behavior of task-related objects. Meaning that robots have to predict and consider the physical effects of their possible actions to take.

In this work we investigate a simulation-based approach to naive physics temporal projection in the context of autonomous robot everyday manipulation. We identify the abstractions underlying typical first-order axiomatizations as the key obstacles for making valid naive physics predictions. We propose that temporal projection for naive physics problems should not be performed based on abstractions but rather based on detailed physical simulations. This idea is realized as a temporal projection system for autonomous manipulation robots that translates naive physics problems into parametrized physical simulation tasks, that logs the data structures and states traversed in simulation, and translates the logged data back into symbolic time-interval-based first-order representations. Within this paper, we describe the concept and implementation of the temporal projection system and present the example of an egg-cracking robot for demonstrating its feasibility.

Categories and Subject Descriptors

I.2 [Artificial Intelligence]: Knowledge Representation Formalisms and Methods

General Terms

Design, Experimentation

Keywords

Temporal Projection, Naive Physics, Simulation, Cognitive Robots

1 Introduction

Accomplishing everyday manipulation tasks successfully requires robots to predict the consequences of actions before committing to them: the robot has to decide where and how hard to hit an egg in order to open it without damaging its content (see Figure 1). Or, it should reason about whether it is necessary to hold a cup upright to avoid spilling the coffee inside. To make such decisions the robot has to predict the changes of the physical state caused by its actions.

To compute the consequences of picking up a cup of coffee too rapidly we can model the setting as a fluid dynamics problem and

Cite as: Simulation-based Temporal Projection of Everyday Robot Object Manipulation, Lars Kunze, Mihai Emanuel Dolha, Emitza Guzman, and Michael Beetz, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 107-114. Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

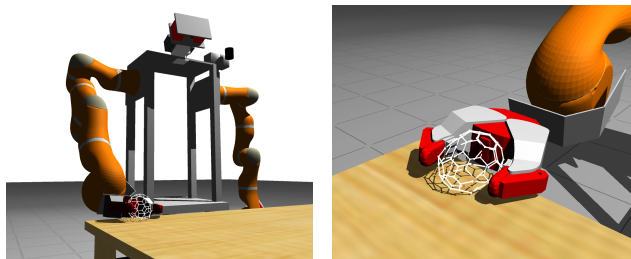


Figure 1: Robot TUM-Rosie manipulates egg in simulation.

solve the respective equations for the variables of interest. This way the robot could compute the fluid flow caused by the robot action and derivated variables. However, these computations do not readily provide the information needed to choose the appropriate action parametrization. It is more informative to predict whether or not a given action parametrization will cause coffee to be spilled. Abstracting the reality into a small qualitative state space, such as coffee spilled or not spilled will also cut down the search space for action selection and thereby make the search more tractable.

Researchers in Artificial Intelligence have investigated approaches to represent and reason about such knowledge under the notion of *naive physics* and *commonsense reasoning*. The attempt to formulate and automate this knowledge using first-order logic has received most of the attention so far. Researchers in qualitative reasoning [18] have formalized various physics problems. The objectives of this approach are most comprehensively stated in Hayes's Naive Physics Manifesto [8]. More recently, the Common Sense Problem Page [14] lists challenge problems. Most relevant are attempts to so-called mid-size axiomatizations [15].

The basic idea is to formalize the laws of physics and situations as logical axioms in an abstract and qualitative language and then deduce the predictions of what will happen from these axiomatizations. Unfortunately, the formalizations tend to become very lengthy and often it is difficult to make the right predictions based on axiomatizations of qualitative physics. One of the main reasons is that logical axiomatizations often quantify over the values of state variables or abstract away from some state variables assuming that they are not relevant for valid predictions. However, when considering actions such as cracking an egg the effects of actions can vary largely with small changes of action parametrizations. The effects depend on where exactly and how hard the egg is hit, how strong and where it is held, and on the exact state of the egg's yolk, etc. Without exactly knowing the values of all state variables it might be impossible to predict the action effects.

Indications of these difficulties are the number and the restrictions of action logics that try to capture phenomena such as concurrent actions, the size of axiomatizations of simple physical phe-

nomena for problems on the Common Sense Problem Page, or, the impossibility to perform certain predictions in a qualitative representations, such as predicting whether a robot will see a certain object when it navigates through the environment while at the same time turning its camera.

These problems do not occur in physics simulations where physics engines (such as ODE¹ or Bullet²) can simulate such phenomena without problems because they apply accurate dynamics models at a fine level of granularity.

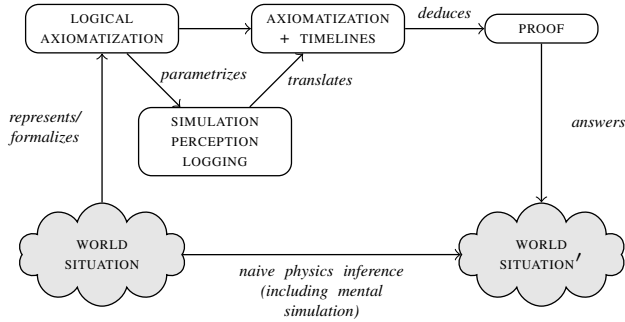


Figure 2: Naive physics inference scheme.

In this paper we combine the ideas of qualitative reasoning about courses of action and their physical effects and having accurate and realistic modeling as depicted in Figure 2. We do so by translating qualitative physics problem formalizations into a parametrized simulation problem, performing a detailed physics-based simulation, logging the state evolution into appropriate data structures and then translating these subsymbolic data structures into an interval-based first-order symbolic/qualitative representation of the respective episode. The resulting fact-base is then used to infer the answers to the qualitative reasoning problems.

The key contribution of this paper is the combination of first-order symbolic representation with physics-based simulation as an inference mechanism for predicting the effects of actions. This combination provides the best of both worlds: it provides the structure and compactness of symbolic representations *and* the realism and accuracy of physics-based simulation. Taken together the robot can predict consequences of actions such as whether an egg will break when the robot performs a specific parametrized move, whether a table will be clean after wiping it with a sponge or whether the sponge needs to be pressed out before, whether using a specific parametrization of a pick up action would cause the coffee in a cup be spilled. The point is that while these predictions are symbolic they are computed from realistic models. Combining the simulation with sampling in the state space as well as in the parametrization space of actions also allows for probabilistic predictions.

In the remainder of the paper we proceed as follows: First, we shortly revisit a well-known problem in naive physics, namely *cracking an egg*. Second, we explain how our approach addresses problems of this kind by tightly integrating logic-based reasoning and physics-based simulation. Third, we demonstrate the feasibility of our approach through experiments. Finally, we conclude after discussing related work.

2 Cracking An Egg

In this paper we take the cracking of an egg as our running example. Egg cracking has been proposed by [3] as a challenge problem for logical formalization and reads as follows:

“A cook is cracking a raw egg against a glass bowl. Properly performed, the impact of the egg against the edge of the bowl will crack the eggshell in half. Holding the egg over the bowl, the cook will then separate the two halves of the shell with his fingers, enlarging the crack, and the contents of the egg will fall gently into the bowl. The end result is that the entire contents of the egg will be in the bowl, with the yolk unbroken, and that the two halves of the shell are held in the cook’s fingers.”

Solutions to this problem should not only characterize aspects mentioned above but also account for variants of the problem:

“What happens if: The cook brings the egg to impact very quickly? Very slowly? The cook lays the egg in the bowl and exerts steady pressure with his hand? The cook, having cracked the egg, attempts to peel it off its contents like a hard-boiled egg? The bowl is made of looseleaf paper? of soft clay? The bowl is smaller than the egg? The bowl is upside down? The cook tries this procedure with a hard-boiled egg? With a coconut? With an M & M?”

The cracking an egg problem poses many challenges, especially in the context of everyday robot manipulation. In order to solve it we regard the following aspects to be substantial: First, the abstraction level of a formalization should reflect the sensing and acting capabilities of the manipulating robot. Second, variants should be handled without the need of explicit modeling. And third, concurrent actions and events should be taken into account.

3 Temporal Projection

Let’s now consider how simulation-based temporal projection infers answers to naive physics problems like cracking an egg or pouring coffee to a cup. After giving a short overview of the overall system, we present each part in more detail.

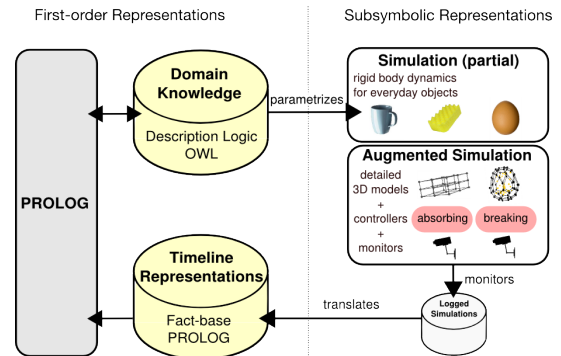


Figure 3: Simulation-based temporal projection system.

3.1 System Overview

An overview of the proposed simulation-based temporal projection system is shown in Figure 3. We formalize task-relevant domain knowledge about object classes and individuals within an ontology. Assertions about individual objects are stored in a knowledge base and used for automatically parametrizing a physics-based simulation. Within the simulator we describe everyday objects with detailed 3D models, augment the descriptions with object controllers that compute physical phenomena not covered by the rigid body dynamics, e.g. breaking of objects, and attach monitoring routines to objects in order to collect object specific data. For a task like egg

¹<http://www.ode.org>

²<http://www.bulletphysics.com>

cracking, we run simulations with differently parametrized control programs, whereby states of objects are monitored and logged. Logged simulations are then translated into time-interval-based representations, called timelines. Finally, we use PROLOG for reasoning about the generated timelines. As can be seen in Figure 3, for its reasoning PROLOG also accesses the domain knowledge.

3.2 Domain Knowledge

We use first-order representations to formalize domain knowledge. Within our approach, we describe general physical knowledge about object types and their properties as well as specific knowledge about individuals in Description Logic (DL). In the following we explain what kind of knowledge we represent.

For representing domain knowledge in DL, we use the semantic web ontology language OWL³. We build our representations on OpenCyc's⁴ upper-ontology and extend type and property descriptions whenever necessary. For example, let's have a closer look at the physically relevant knowledge about eggs and how it can be formalized in DL. We consider an egg as consisting of an eggshell and its content, i.e. egg white and egg yolk. An eggshell is a solid rigid (but fragile) container that has a shape, a mass, and extensions in space. Since the eggshell is fragile it can break. The egg's content is a liquid which has a viscosity and a mass. Figure 4 depicts a simplified excerpt of the ontology that shows type, relation and property information about eggs. For describing a specific situation individuals of relevant objects and their properties are explicitly asserted, e.g., an individual of type *Egg*, *egg3*, has *eggshell3* and *yolk3* as its parts, where *eggshell3* and *yolk3* have a mass of 0.01 and 0.04 respectively. Other properties and relations are specified similarly. For a specific task like egg cracking information about all relevant objects is asserted in the knowledge base. These assertions build the basis for parametrizing the physics-based simulation which is explained in the next section.

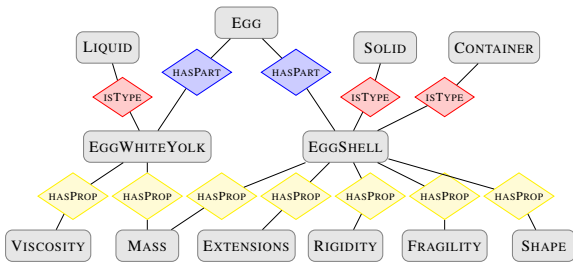


Figure 4: Ontology showing physical aspects of eggs.

3.3 Physics-based Simulation

Within our approach, we utilize a physics-based simulator, namely Gazebo⁵, for computing the effects of robot actions, object interactions and other physical events.

For the computation, we parametrize the simulator on the basis of the logical axiomatization, i.e. the domain knowledge, run simulations and log data of features like position, velocity, forces, and contact points between objects over time. After explaining shortly how a physics-based simulator computes physical effects generally, we present how the Gazebo simulator can be configured and how we derive a configuration based on the assertions in the knowledge base.

Generally a physics-based simulator works as follows: the simulator starts its computation of physical effects based on an initial

configuration. Then it periodically receives motor control commands which are translated into forces and updates the state of the simulated world according to physical laws. Within each tiny update step, forces are applied to affected objects by considering both the object's current dynamic state and its properties like mass and friction. Later we explain how we augment the simulation in order to account for physical phenomena like breaking or absorbing.

The initial configuration of the Gazebo simulator is based on an XML file, called *world file*. The world file describes properties of the simulation, specifies parameters for the physics engine (ODE) and describes all things occurring in the world, including robots, sensors and everyday objects. The following excerpt of a world file shows entries for the physics engine and the objects *eggshell3* and *yolk3*.

```
<gazebo:world ...>
  <physics:ode>
    <stepTime>.006</stepTime>
    <gravity>0 0 -9.8</gravity>...
  </physics:ode>
  <model:physical name="eggshell3">
    <xyz>0 0 1.23</xyz>
    <rpy>0 0 0</rpy>
    <include embedded="true">
      <xi:include href="../models/eggshell3.model" />
    </include>
  </model:physical>
  <model:physical name="yolk3">...
</gazebo:world>
```

Within a world file each object has its own model description. Such model descriptions comprise mainly the object's shape and a set of physical properties like size, mass, and rigidity. Figure 5 visualizes some parameters for both models: *eggshell3* and *yolk3*. These models configurations are derived from the information stored in the knowledge base. When properties are not explicitly specified within the knowledge base, we simply assume default values.

To simulate physical phenomena like breaking objects we augment the model descriptions, how this is realized is presented in the next section.

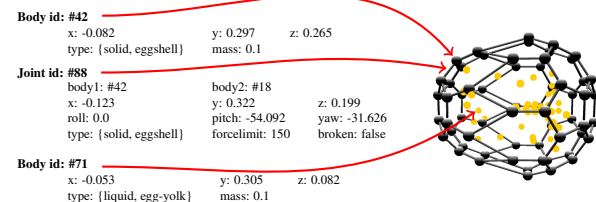


Figure 5: Modeling shape and physical properties of an egg. The shape of the egg is modeled with a graph-based structure of bodies which are linked by joints. The physical properties of these individual bodies and joints, which are shown exemplarily on the left side, determine the physical properties of the whole egg, e.g. its mass and fragility.

3.4 Augmented Simulation

The Gazebo simulator is designed for simulating robots, sensors and objects, whereby physical aspects of objects and their interactions are more or less limited to rigid body dynamics. Since we want to simulate naive physics problems with phenomena like breaking, cutting, mixing, cooking, baking, or melting we augment object model descriptions with detailed shape models, controllers for simulating physical phenomena, and monitors for logging states of objects. The extended model descriptions are collected in a library for simulating phenomena of everyday physics.

³<http://www.w3.org/2004/OWL>

⁴<http://www.opencyc.org>

⁵<http://playerstage.sourceforge.net/gazebo/gazebo.html>

Instead of modeling objects as rigid bodies, we describe the shape of objects similar to work by [9] with graph-based structures which allow us to inspect physical aspects at a more detailed level. Figure 5 visualizes the shape of an eggshell with egg yolk inside. The basic entities for modeling the shape of an object are *bodies* and *joints*, which are mutually connected. Properties of an object like type, mass, spatial extensions, and rigidity determine the attributes of these basic entities.

In order to simulate new classes of objects, e.g. objects that are breakable, cuttable, or objects that absorb liquids we add controllers to the object model descriptions. These controllers are called within each simulation step and perform some specialized computation. The computation can be based on physical properties calculated by the simulator or on results computed by other controllers. Thereby object attributes like temperature, being wet, being dirty, or being broken can be computed. This allows us to simulate a new range of processes like filling, cutting, or breaking.

In addition to controllers, we add monitoring routines to object model descriptions to log the object’s state at each simulation step. The data that is monitored and logged is specified for each object individually.

3.5 From Logged Simulations to Timelines

In this section we explain how we ground first-order representations in logged data structures of the simulator. Before we explain how log files are translated into logic, we will present the representation formalism for temporal knowledge and shortly discuss its relation to domain knowledge.

For representing temporal knowledge, i.e. object configurations and events at given time points, we make use of notations common in the *event calculus* [10] and its extensions. In the following we present predicates relevant for temporal reasoning.

The notation is based on two concepts, namely fluents and events. Fluents are conditions that change over time, e.g., a cup contains coffee: *contains(cup,coffee)*. Events (or actions) are temporal entities that have effects and occur at specific points in time, e.g., consider the action of pouring coffee: *pourTo(coffee,pot,cup)*. Logical statements about both fluents and events are expressed mainly by two predicates:

- *Holds(f,t)* and
- *Occurs(ev,t)*,

where *f* denotes a fluent, *ev* denotes an event and *t* simply denotes a point in time. The statement *Holds(f,t)* represents that fluent *f* holds at time *t*, whereas *Occurs(ev,t)* represents an occurrence of event *ev* at time *t*. Although fluents and events look as if they were predicates themselves, they are not: both fluents and events are reified as functions returning respective instances. Thus, by treating them as ‘first-class citizens’ in a first-order representation allows us to state at what points in time they hold or occur.

The relation of domain and temporal knowledge is straight forward. Domain knowledge, in particular the assertions about individual objects, characterize the initial conditions for the temporal reasoning. From the temporal reasoning point of view, the assertional knowledge holds at time point *0.0*, i.e. *Holds(f,0.0)*. That the assertional knowledge describes the initial conditions for the temporal reasoning perfectly makes sense since it is also used for parametrizing (or initializing) the simulation as we explained earlier.

Logged simulations are translated into interval-based timeline representations by using the predicates *Holds* and *Occurs*. Whenever a fluent or event is recognized an instance of its corresponding type is generated and either the *Holds* or the *Occurs* predicate is asserted for the observed timepoint. We reuse a generated instance

only if the fluent or event is also valid in successive timesteps. Thereby we get an interval-based representation of timelines. Table 1 and Table 2 list examples of implemented fluents and events for which we assert predicates from the logged simulations.

Table 1: Fluents for static physical configurations.

fluent	intuitive description
<i>contacts(o₁, o₂)</i>	object <i>o₁</i> and object <i>o₂</i> contact each other
<i>attached(o₁, o₂)</i>	object <i>o₂</i> is attached to object <i>o₁</i>
<i>supports(o₁, o₂)</i>	object <i>o₁</i> supports object <i>o₂</i>
<i>contains(o₁, o₂)</i>	container <i>o₁</i> contains object (or stuff) <i>o₂</i>
<i>broken(o₁)</i>	object <i>o₁</i> is broken
<i>spilled(o₁)</i>	object <i>o₁</i> is spilled

Table 2: Fluents for physical events.

fluent	intuitive description
<i>colliding(o₁, o₂)</i>	object <i>o₁</i> and object <i>o₂</i> are colliding
<i>falling(o₁)</i>	object <i>o₁</i> is falling
<i>moving(o₁)</i>	object <i>o₁</i> is moving
<i>openingGripper(o₁)</i>	robot is opening gripper <i>o₁</i>
<i>closingGripper(o₁)</i>	robot is closing gripper <i>o₁</i>
<i>breaking(o₁)</i>	object <i>o₁</i> is breaking
<i>spilling(o₁)</i>	object <i>o₁</i> is spilling over a surface

How fluents and events are grounded in the data structures of the simulator is exemplarily explained for the fluents *contacts(o₁, o₂)* and *supports(o₁, o₂)* and the events *moving(o₁)* and *breaking(o₁)*.

A contact between objects is directly reported by the simulator:

$$\begin{aligned} \text{Holds}(\text{contacts}(o_1, o_2), t_i) &\Leftrightarrow \\ \text{Collisions} &= \text{SimulatorValueAt}(\text{Collisions}, t_i) \wedge \\ \text{Member}((o_1, o_2), \text{Collisions}) & \end{aligned}$$

Object *o₁* supports an object *o₂* when there exists a contact between both objects and the maximum value of *o₁*’s bounding box within *z*-dimension is slightly less or equal than the minimum value of *o₂*’s bounding box and *o₂*’s center of mass lies within the spatial extensions of object *o₁* regarding the *x*-*y*-dimensions. The later condition is captured by the *isDirectlyBelow* predicate. Furthermore the gravity force of *o₂* has to be canceled out:

$$\begin{aligned} \text{Holds}(\text{supports}(o_1, o_2), t_i) &\Leftrightarrow \\ \text{Holds}(\text{contacts}(o_1, o_2), t_i) &\wedge \\ p_1 &= \text{SimulatorValueAt}(\text{Pose}(o_1), t_i) \wedge \\ p_2 &= \text{SimulatorValueAt}(\text{Pose}(o_2), t_i) \wedge \\ \text{isDirectlyBelow}(p_1, p_2) &\wedge \\ \text{gravityForceIsCanceledOut}(o_2) & \end{aligned}$$

An object *o₁* is moving when its pose has changed between two successive timesteps *t_j* and *t_i*:

$$\begin{aligned} \text{Occurs}(\text{moving}(o_1), t_i) &\Leftrightarrow \\ p_1 &= \text{SimulatorValueAt}(\text{Pose}(o_1), t_i) \wedge \\ p_2 &= \text{SimulatorValueAt}(\text{Pose}(o_1), t_j) \wedge \\ \text{previousTimestep}(t_j, t_i) &\wedge \\ p_1 &\neq p_2 \end{aligned}$$

An object *o₁* is breaking in timestep *t₁* when one of its joints is detached within that timestep. The controller that realizes the breaking phenomenon of objects directly reports which joints are detached in a timestep:

$$\begin{aligned} \text{Occurs}(\text{breaking}(o_1), t_i) &\Leftrightarrow \\ j_1 &= \text{SimulatorValueAt}(\text{Detached}(joint_1), t_i) \wedge \\ \text{Member}(j_1, \text{GetJoints}(o_1)) & \end{aligned}$$

The next section explains how we do reasoning on the grounded fluents and events asserted in timelines.

3.6 Reasoning on Timelines

Figure 6 shows a sequence of images of a simulated egg dropped onto the floor. By examining this simple example we show how PROLOG can be used for reasoning about both timelines derived from logged simulations and domain knowledge. Let’s consider the PROLOG query:

```
?- holds(F1,T1), fluentT(F1,supports), objOf(F1,egg1),
   after(T2,T1), occurs(E1,T2), objOf(E1,egg1).
```

where $F1$ and $E1$ are variables for a fluent and event respectively, $T1$ and $T2$ are time intervals, $supports$ is the type of fluent $F1$, and $egg1$ denotes an individual. The query basically asks for all events $E1$ that hold for $egg1$ after $egg1$ is no longer supported. For the dropped egg example, $E1$ is bound to $falling(egg1)$, $colliding(egg1,floor)$, and $breaking(egg1)$. The *after* relation used in the query above is one of the thirteen possible temporal relationships between time intervals [1] which we have implemented as predicates for reasoning about timelines.

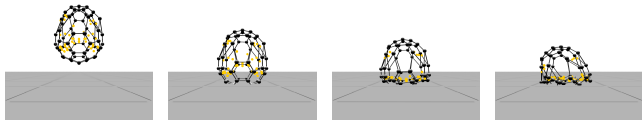


Figure 6: Simulation of an egg dropped onto the floor. From left to right: falling, colliding, breaking, and broken.

Figure 7 illustrates the complete process of naive physics reasoning for a situation where a robot wants to grasp an egg but his hand is wet. Similar to the example query above, the robot could ask what will happen if (after) it grasps the egg with its wet hand. The initial conditions describing the actual situation are taken from the knowledge base to parametrize the simulation, the fact that the hand is wet would reduce the friction of the hand. Then the simulation is run whereby states of objects are monitored and logged. After the logged simulations are translated into interval-based first-order representations the query will be answered based on the resulting timelines. Depending on the reduced friction of the hand the egg might slip away and fall onto the floor which cause the eggshell to break and the egg yolk to be spilled.

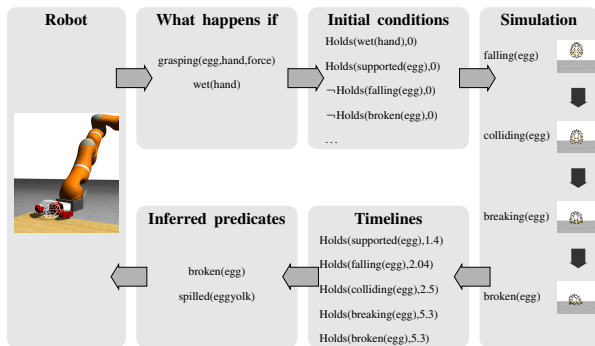


Figure 7: Complete process of the temporal projection.

In addition to fluents and events that are grounded within the data structures of the simulator, more complex events like picking up an object or an overflowing container can be defined by utilizing grounded fluents, events and additional temporal constraints. For the description of complex events we follow the notion of chronicles [7].

4 Experiments

For showing the feasibility of our approach we have conducted several robot manipulation experiments for the problem of cracking an egg as described in Section 2. In these experiments we addressed the requirements posed in the problem formulation. Furthermore we have conducted experiments for the problem of pouring and absorbing liquids.

The robot model used in our experiments is the PR2 robot platform developed by Willow Garage⁶. The PR2 has an omnidirectional base, a telescoping spine and a pan-tilt head. Each of the two compliant arms of the platform have four degrees of freedom (DOF) with an additional three DOF in the wrist and one DOF gripper. The sensor setup is comprised of a laser sensor on the base, a tilting laser sensor for acquiring 3D point clouds, two stereo camera setups and a high resolution camera in the head. The hands also have cameras in the forearms, while the grippers have three-axis accelerometers and fingertip pressure sensor arrays. The entire setup is realistically modeled and ready to use in the Gazebo simulator.

4.1 Cracking an Egg

Cracking an egg against another object and then separating (splitting) it requires a robot to be able to grasp an egg at all. Therefore we start our experiments with a scenario where a robot is supposed to simply grasp an egg lying on a table.

The first experiment consists of several trials in which a robot, in this case the PR2, is using different values of gripper force to grasp an egg. The experiment underlines the importance of a physical simulation since it allows to determine an appropriate force for grasping an egg which would not be possible by pure symbolic reasoning.

The simulation setup for this experiment is simple and consists of the PR2 robot model and the egg model lying on a table being spawned in a Gazebo environment. The robot is trying to pick up the egg by applying different forces with his gripper. It starts with the lowest force level and after each try the force is increased, the old egg is unspawned and a new one is created as the old one might be damaged during the experiment. In each trial the robot tries to pick the egg up and hold it up for a period of time. During the experiments we found three possible outcomes (see Figure 8): the egg slips out of the robot’s gripper, the egg is held by the robot successfully, and the egg is crashed by the robot (Figure 9).

Within the experiment we identified four force levels as being too low for grasping the egg, three force levels as appropriate and five force levels as being too high. A video showing this experiment is available online⁷.

The data structures of the simulation were logged and translated into interval-based first-order representations. Thereby the results of the experiment are made available in the logical programming environment PROLOG which is demonstrated by the following queries

```
?- occurs(E,T1), eventT(E,'ClosingGripper'),
   argsOf(E, [rightGripper, force7]), holds(F,T2),
   after(T2,T1), fluent(F,Type), argsOf(F,Args).
```

```
E = closingGripperEvt7,
T1 = 3.25,
F = attachedF11,
T2 = 5.0,
Type = 'Attached',
Args = [rightGripper, egg1].
```

```
?- occurs(E,T1), eventT(E,'ClosingGripper'),
   argsOf(E, [rightGripper, force12]), holds(F,T2),
```

⁶<http://www.willowgarage.com/pages/pr2/overview>

⁷<http://www.youtube.com/watch?v=MzMnTooXyCc>

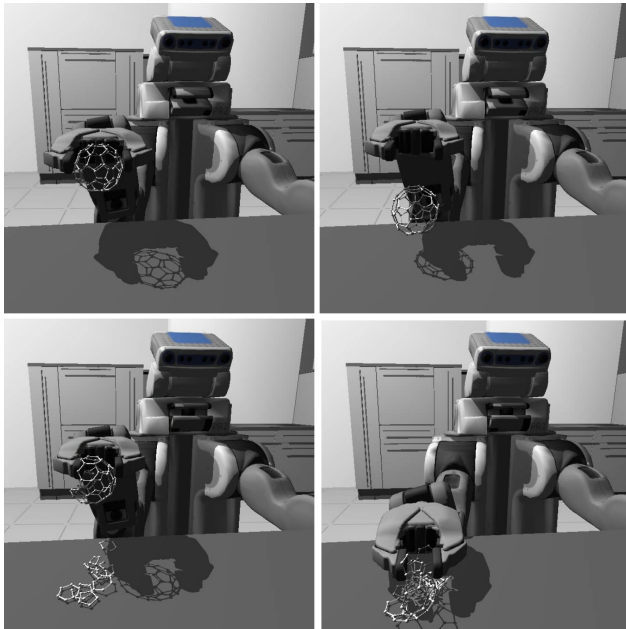


Figure 8: PR2 robot picking up an egg with different force levels (upper left: successful; upper right: egg slipping; bottom left: egg crashed, parts of the eggshell fell onto the table; bottom right: egg crashed).

```
after(T2,T1), fluent(F,Type), argsOf(F,Args).
```

```
E = closingGripperEvt12,
T1 = 3.0,
F = brokenF11,
T2 = 4.25,
Type = 'Broken',
Args = [egg1]
```

where the first and the second query ask for fluents that hold after grasping the egg with *force7* and *force12*, respectively. Whereas using *force7* result in a successful trial where the egg is attached to the robot’s gripper, using *force12* result in a situation where the egg is broken. Given the simulation-based temporal projections of what will happen if the robot grasps an egg with a particular force it is possible to determine an appropriate value for the grasp force parameter.

In the second experiment we used a valid grasping force to pick up an egg and test its behavior when hitting it against obstacles and tables. The egg is picked up and then is hit or pressed against an obstacle. The results here are of course dependent on the forces that affect the egg model: while hitting the egg against another object very gently would not break it, hitting it stronger or pressing it firmly against the table would produce breaking. Figure 10 shows the egg model being cracked after being hit against an obstacle. This experiment can be used to gather information on how to safely manipulate such a fragile object and how the robot’s actions influence the forces applied to the object.

The last experiment was focused on egg splitting. The robot is grasping the egg from the table that’s lying on the table and, after hitting it against an obstacle and cracking it, is trying to split it using his other gripper (Figure 11). This experiment was not entirely successful as the egg to be too fragile for the PR2 grippers. The result of most of the trials involved in this experiment was the cracked egg being completely crushed by the two grippers.

In contrast to the logical formalization that has been proposed

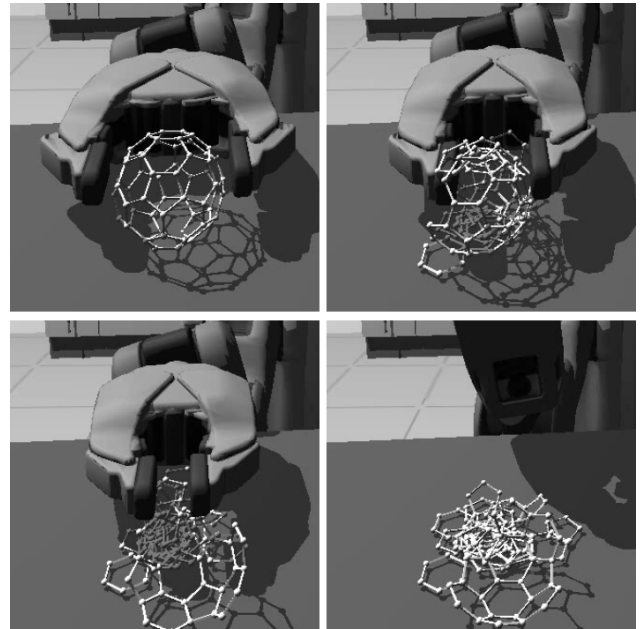


Figure 9: The egg model crashing in a grasp trial because of too much gripper force.

by [15] our approach is able to make temporal projections about almost all aspects of the problem specification and its variants. Their theory [15] is based on roughly 70 axioms, but variations such as

- the cook brings the egg to impact very quickly or very slowly
- the bowl is upside down
- the cook tries the procedure with a hard-boiled egg, coconut, or an M&M
- the cook puts the egg in the bowl and exerts steady pressure with his hand
- the cook, having cracked the egg, attempts to peel it off its contents like a hard-boiled egg

cannot be handled without further extensions. All these variations, except the last, seem to be feasible with our simulation approach. We simply have to adapt the robot control program to induce a different manipulation behavior, to change the configu-

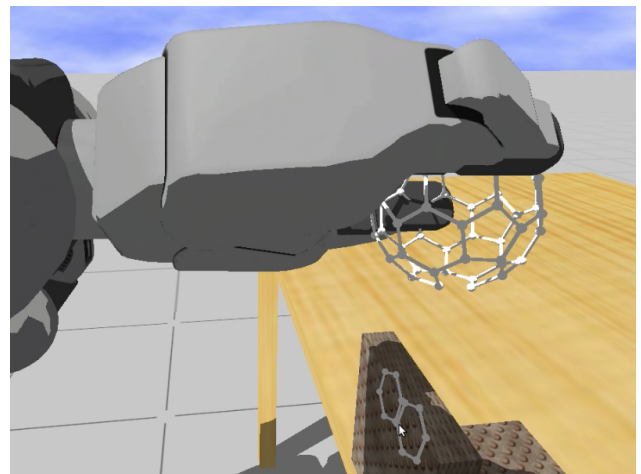


Figure 10: An egg model cracked by hitting an obstacle.

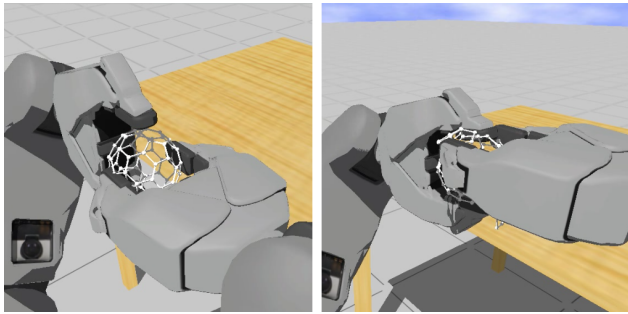


Figure 11: An egg splitting trial using the PR2 robot model.

ration of the environment, or to adjust the physical parameters of the object models, e.g. size, structure and/or fragility of objects. Although the adjustment of the physical parameters is not trivial, it seems to be much easier than the extension of a logical theory since machine learning techniques can be applied for finding the appropriate physical models.

Figure 12 shows the robot moving its arm away from the egg after breaking it. In the beginning, parts of the eggshell stuck to the gripper and fell off at a later point in time. It is impossible to model such phenomena within logical abstractions, whereas in detailed simulations they simply emerge from the laws of physics. This example strongly emphasizes the benefit of combining first-order representations with physics simulations.

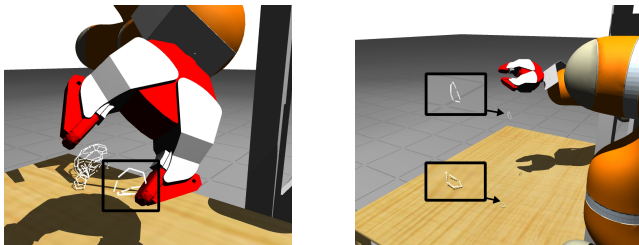


Figure 12: Grasping an egg. (1) eggshell stuck to gripper (2) eggshell fell off the gripper.

4.2 Pouring Liquids

In this scenario the robot picks up a filled container from the table, moves it above a second container, and then pours the water from the first to the second container. We looked at several variants of the problem: (a) the second container has holes, (b) the second container is upside down, (c) honey (instead of water) is poured to an upside down container (d) the task is performed successfully, meaning that the liquid completely ends up in the second container, (e) the second container is already filled which causes it to overflow as soon as the robot pours further liquid to it, meaning that some liquid is spilled on the table.

The following query asks for the conditions that hold after the pouring action:

```
?- occurs(E, T1), eventT(E, pouringTo),
  argsOf(E, [cup1, liq1, cup2]),
  after(T2, T1), holds(F, T2).
```

where in variant (a) and (e) F is bound to $contains(cup2, liq1)$, $spilled(liq1)$ and $supports(table1, liq1)$, in variant (b) F is bound to $spilled(liq1)$, $supports(table1, liq1)$ and $supports(cup2, liq1)$, in (c) F is bound to $spilled(liq1)$ and $supports(cup2, liq1)$, and in (d) F is bound to $contains(cup2, liq1)$ (Figure 13). The notable difference between variants (b) and (c) result from the fact that honey has a

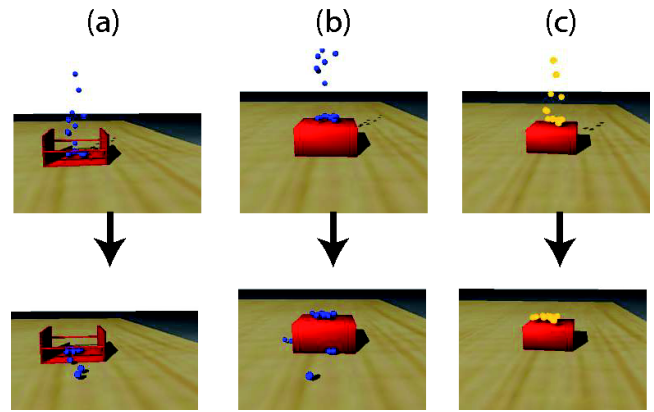


Figure 13: Pouring liquids to containers (a) Pouring liquid to container with holes (b) Pouring water to an upside down container (c) Pouring honey to an upside down container.

higher viscosity than water. Within the simulation this is reflected by the different friction values for water and honey.

In a follow-up experiment the robot cleans the table with a sponge (Figure 14). The corresponding query looks as follows:

```
?- occurs(E, T1), eventT(E, wiping),
  argsOf(E, [sponge1, table1]),
  after(T2, T1), holds(F, T2).
```

where F is bound to $absorbs(sponge1, liq1)$.

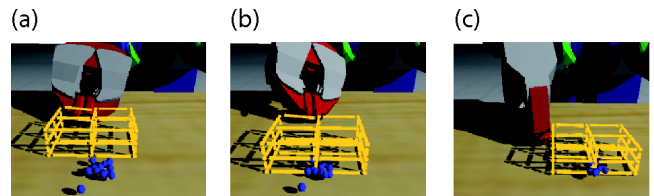


Figure 14: Wiping table with sponge (teleoperated) (a) Gripper approaches sponge (b) Gripper pushes sponge over liquid which in turn is absorbed (c) Gripper pushes sponge and both absorbed liquid and sponge move together.

5 Related Work

Solutions to a naive physics problem, namely egg cracking [3], were formulated by [11, 15] based on logical axiomatizations. Limitations of these approaches are mainly that physical details are abstracted away and that variants cannot be handled very flexibly. To overcome such limitations this work proposes a simulation-based approach: we take a logical axiomatization and translate it into a parametrized simulation problem, simulate and log simulation data, translate logged simulation data into an interval-based first-order representation which is used for answering queries about a qualitative reasoning problem.

The integration of numerical simulation and qualitative methods has been investigated before, for example, work on qualitative-numeric simulation [2] and self-explanatory simulations [5]. Work by [12] has shown an integration of numerical simulation and qualitative modeling based on the Qualitative Process Theory [4] for virtual interactive environments. But none of the approaches, we are aware of, have investigated a simulation-based approach for making predictions in the context of every robot object manipulation.

Our simulation-based approach is in a similar line of work by [9] who integrated logic and simulation for commonsense reasoning. Whereas they use a general purpose simulation, we utilize a physics-based simulator augmented with phenomena of everyday physics since we are particularly interested in naive physics reasoning for robot manipulation. Instead of looking at isolated problems, we aim for a tight integration between the our proposed reasoning system and other processes like planning, e.g., to predict whether a meal is edible when executing a specific plan for cooking pasta.

The grounding of logical predicates like *contacts*(o_1, o_2) in data of logged simulations is done similar to work by [17] who grounded semantics in visual perception. Similarly, we ground only primitive predicates in logged simulations. Complex predicates are formulated in PROLOG and are based on primitive or other complex predicates similar to definitions of symbolic chronicles [7].

The underlying idea of our approach is not restricted to problems in naive physics, but it can effectively be applied to tasks like the visibility of objects in scenes, perspective taking [13], physics-based motion planning [19], navigation in environments with non-rigid objects [6], and the prediction of interfering effects of continuous and concurrent actions [16].

6 Conclusions

In this paper we presented a simulation-based approach to naive physics temporal projection in the context of robot manipulation. Instead of making predictions based on logical axiomatizations, we propose an inference system based on physics simulations.

We developed techniques for transferring qualitative reasoning problems to physics simulations, for monitoring states of objects and logging them to appropriate data structures, for translating logs into first-order representations, and for answering queries on the resulting representations. We successfully conducted experiments that show that it is feasible to infer answers to naive physics problems based on physical simulations. We think that the effort needed to get a functionality in simulation-based inference is less than for axiomatizing the respective functionality. Additionally, details of investigated problems are not abstracted away within detailed simulations. Furthermore, inference tasks that are notoriously difficult to axiomatize are doable in simulation-based reasoning, for example, tasks including concurrent actions, soft bodies, liquids, and physical processes like mixing, overboiling, or scorching. In this work we are neither aiming for high-performance nor high-fidelity simulations, but rather exploiting simulation technologies for answering questions about naive physics problems which are abstracted in a reasonable small qualitative state space. We expect that issues related to performance and realistic models will be addressed by the game and animation film industry.

Thereby, we believe that this system provides a functionality needed for successfully accomplishing complex everyday robot manipulation tasks. In future work, we will address how robots can employ the simulation-based temporal projection approach for determining the appropriate parametrizations for their actions.

7 Acknowledgments

This work is supported in part within the DFG excellence initiative research cluster *Cognition for Technical Systems – CoTeSys*⁸.

8 References

- [1] J. F. Allen. Maintaining knowledge about temporal intervals. In *Readings in qualitative reasoning about physical systems*, pages 361–372. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.
- [2] D. Berleant and B. Kuipers. Qualitative-numeric simulation with q3. In *RECENT ADVANCES IN QUALITATIVE PHYSICS*, pages 3–16. The MIT Press, 1992.
- [3] E. Davis. Cracking an Egg. Common Sense Problem Page, <http://www-formal.stanford.edu/leora/commonsense/#eggcracking>, 9 1997.
- [4] K. D. Forbus. Qualitative process theory. *Artif. Intell.*, 24(1-3):85–168, 1984.
- [5] K. D. Forbus and B. Falkenhainer. Self-explanatory simulations: An integration of qualitative and quantitative knowledge. In *AAAI*, pages 380–387, 1990.
- [6] B. Frank, C. Stachniss, R. Schmedding, M. Teschner, and W. Burgard. Real-world robot navigation amongst deformable obstacles. In *ICRA'09: Proceedings of the 2009 IEEE international conference on Robotics and Automation*, pages 617–622, Piscataway, NJ, USA, 2009. IEEE Press.
- [7] M. Ghallab. On Chronicles: Representation, On-line Recognition and Learning. In *Principles of Knowledge Representation and Reasoning-International Conference-*, pages 597–607. Morgan Kaufmann Publishers, 1996.
- [8] P. Hayes. The second naive physics manifesto. In J. R. Hobbs and R. C. Moore, editors, *Formal Theories of the Commonsense World*, pages 1–36. Ablex, Norwood, NJ, 1985.
- [9] B. Johnston and M. Williams. Comirit: Commonsense Reasoning by Integrating Simulation and Logic. In *Artificial General Intelligence 2008: Proceedings of the First AGI Conference*, page 200. IOS Press, 2008.
- [10] R. Kowalski and M. Sergot. A logic-based calculus of events. *New Gen. Comput.*, 4(1):67–95, 1986.
- [11] V. Lifschitz. Cracking an Egg: An Exercise in Commonsense Reasoning. Presented at the 4th Symposium on Logical Formalizations of Commonsense Reasoning, 1998.
- [12] J.-L. Lugrin and M. Cavazza. Making sense of virtual environments: action representation, grounding and common sense. In *IUI '07: Proceedings of the 12th international conference on Intelligent user interfaces*, pages 225–234, New York, NY, USA, 2007. ACM.
- [13] L. Marin, E. A. Sisbot, and R. Alami. Geometric tools for perspective taking for human-robot interaction. In *MICAI 2008*, 2008.
- [14] R. Miller and L. Morgenstern. Common Sense Problem Page. <http://www-formal.stanford.edu/leora/commonsense>, 2009.
- [15] L. Morgenstern. Mid-sized axiomatizations of commonsense problems: A case study in egg cracking. *Studia Logica*, 67(3):333–384, 2001.
- [16] L. Mösenlechner and M. Beetz. Using physics- and sensor-based simulation for high-fidelity temporal projection of realistic robot behavior. In *ICAPS 2009*, 2009.
- [17] J. M. Siskind. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *J. Artif. Intell. Res. (JAIR)*, 15:31–90, 2001.
- [18] D. S. Weld and J. d. Kleer, editors. *Readings in qualitative reasoning about physical systems*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.
- [19] S. Zickler and M. Veloso. Efficient physics-based planning: sampling search via non-deterministic tactics and skills. In *AAMAS '09: Proceedings of The 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 27–33, Richland, SC, 2009. IFAAMAS.

⁸<http://www.cotesys.org>

Online Anomaly Detection in Unmanned Vehicles

Eliahu Khalastchi¹, Gal A. Kaminka², Meir Kalech¹, Raz Lin²

¹DT Labs, Information Systems Engineering, Ben-Gurion University
Beer Sheva, Israel 84105
eli.kh81@gmail.com, kalech@bgu.ac.il

²The MAVERICK Group, Department of Computer Science, Bar-Ilan University
Ramat-Gan, Israel 52900
{galk,linraz}@cs.biu.ac.il

ABSTRACT

Autonomy requires robustness. The use of unmanned (autonomous) vehicles is appealing for tasks which are dangerous or dull. However, increased reliance on autonomous robots increases reliance on their robustness. Even with validated software, physical faults can cause the controlling software to perceive the environment incorrectly, and thus to make decisions that lead to task failure. We present an online anomaly detection method for robots, that is light-weight, and is able to take into account a large number of monitored sensors and internal measurements, with high precision. We demonstrate a specialization of the familiar Mahalanobis Distance for robot use, and also show how it can be used even with very large dimensions, by online selection of correlated measurements for its use. We empirically evaluate these contributions in different domains: commercial Unmanned Aerial Vehicles (UAVs), a vacuum-cleaning robot, and a high-fidelity flight simulator. We find that the online Mahalanobis distance technique, presented here, is superior to previous methods.

Categories and Subject Descriptors

I.2.9 [Artificial Intelligence]: Robotics

General Terms

Experimentation

Keywords

anomaly detection, Mahalanobis Distance, uncertainty, machine learning, robotics

1. INTRODUCTION

The use of unmanned vehicles and autonomous robots is appealing for tasks which are dangerous or dull, such as surveillance and patrolling [1], aerial search [9], rescue [2] and mapping [19]. However, increased reliance on autonomous robots increases our reliance on their robustness. Even with validated software, physical faults in sensors and actuators can cause the controlling software to perceive the environment incorrectly, and thus to make decisions that lead to task failure.

This type of fault, where a sensor reading can be valid, but invalid given some operational or sensory context, is called *contextual failure* [4].

Cite as: Online Anomaly Detection in Unmanned Vehicles, Eliahu Khalastchi, Gal A. Kaminka, Meir Kalech and Raz Lin, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 115-122.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

tual failure [4]. For instance, a sensor can get physically stuck such that it no longer reports the true value of its reading, but does report a value which is in the range of valid readings.

Autonomous robots operate in dynamic environments, where it is impossible to foresee, and impractical to account, for all possible faults. Instead, the control systems of the robots must be complemented by anomaly-detection systems, that can detect anomalies in the robot's systems, and trigger diagnosis (or alert a human operator). To be useful, such a system has to be computationally light (so that it does not create a computational load on the robot, which itself can cause failures), and detect faults with high degree of both precision and recall. A too-high rate of false positives will lead operators to ignoring the system; a too-low rate makes it ineffective. Moreover, the faults must be detected quickly after their occurrence, so that they can be dealt before they become catastrophic.

In this paper, we focus on online anomaly detection methods for robots. We present methods that are light-weight, and are able to take into account a large number of monitored sensors and internal measurements, with high precision. We make two contributions. First, we argue that in monitoring robots and agents, anomaly detection is improved by considering not the raw sensor readings, but their differential. This is because robots act in the same environment in which they sense, and their actions are expected to bring about changes to the environment (and thus change to their sensor readings). Second, we demonstrate the online use of the Mahalanobis distance—a statistical measure of distance between a sample point and a multi-dimensional distribution—to detect anomalies. The use of Mahalanobis distance is not new in anomaly detection; however, as previous work has shown [12] its use with the high-dimensional sensor data produced by robots is not trivial, and requires determining correlated dimensions. While previous work relied on offline training, to do this, we introduce the use of the lightweight Pearson correlation measure to do this. Taken together, the two contributions lead to an anomaly detection method specialized for robots (or agents), and operating completely on-line.

To evaluate these contributions, we conduct experiments in three different domains: We utilize actual flight-data from commercial Unmanned Aerial Vehicles (UAVs), in which simulated faults were injected by the manufacturer; data from the RV-400 vacuum cleaning robot; and the *Flightgear* flight simulator, which is widely used for research [10, 16, 7]. In all, we experiment with variant algorithms, and demonstrate that the online Mahalanobis distance technique, presented here, is superior to previous methods. The experiments also show that the use of the differential sensor readings improve on competing anomaly detection techniques, and is thus independent of the use of the Mahalanobis distance.

2. RELATED WORK

Anomaly detection has generated substantial research over past

years. Applications include intrusion and fraud detection, medical applications, robot behavior novelty detection, etc. (see [4] for a comprehensive survey). We focus on anomaly detection in Unmanned (Autonomous) Vehicles (UVs). This domain is characterized by a *large amount of data* from many sensors and measurements, that is typically *noisy* and streamed online, and requires an anomaly to be *discovered quickly*, to prevent threats to the safety of the robot [4].

The large amount of data is produced from a large number of system components comprising of actuators, internal and external sensors, odometry and telemetry, that are each monitored at high frequency. The separated monitored components can be thought of as dimensions, and thus a collection of monitored readings, at a given point in time, can be considered a multidimensional point (e.g., [12, 15]). Therefore, methods that produce an anomaly score for each given point, can use calculations that consider the points' density, such as Mahalanobis Distance [12] or K -Nearest Neighbor (KNN) [15]. We repeat such a method here.

When large amounts of data are available, distributions can be calculated, hence, statistical approaches for anomaly detection are considered. These approaches usually assume that the data is generated from a particular distribution, which is not the case for high dimensional real data sets [4]. Laurikkala *et al.* [11] proposed the use of Mahalanobis Distance to reduce the multivariate observations to univariate scalars. Brotherton and Mackey [3] use the Mahalanobis Distance as the key factor for determining whether signals measured from an aircraft are of nominal or anomalous behavior. However, they are limited in the number of dimensions across which they can use the distance, due to run-time issues.

Apart from having to reduce dimensions when using Mahalanobis Distance, the dimensions that are left should be correlated. Recently, Lin *et al.* [12] demonstrated how using an offline mechanism as the Multi-Stream Dependency Detection (MSDD) [14] can assist in finding correlated attributes in the given data and enable use of Mahalanobis Distance as an anomaly detection procedure. The MSDD algorithm finds correlation between attributes based on their values. Based on the results of the MSDD process, they manually defined the correlated attributes for their experiments. However, the main drawback of using the MSDD method is that it consumes many resources and can only be used with offline training. Thus, we propose using a much simpler algorithm, that groups correlated attributes using *Pearson correlation coefficient* calculation. This calculation is both light and fast and therefore can be used online, even on a computationally weak robot.

To distinguish the inherent noisy data from anomalies, Kalman filters are usually applied (e.g., [8, 18, 5]). Since simple Kalman filters usually produce a large number of false positives, additional computation is used to determine an anomaly. For example, Cork and Walker [5] present a non-linear model, which, together with Kalman filters, tries to compensate for malfunctioning sensors of UAVs. We use a much simpler filter that significantly improved the results of our approach. The filter normalizes values using a Z score transformation.

3. ONLINE ANOMALY DETECTION FOR ROBOTS

We begin by describing the problem and outlining our approach. We describe the online training procedure, and the specialization for anomaly detection on robots. Finally, we describe when our approach should flag anomalies and describe our algorithm in detail.

3.1 Problem Description

We deal with the problem of online anomaly detection. Let $A = \{a_1, \dots, a_n\}$ be the set of attributes that are monitored. Monitored attributes can be collected by internal or external sensors (e.g., *odometry, telemetry, speed, heading, GPS_x, GPS_y*, etc.). The data is sampled every t milliseconds. An input vector $\vec{i}_t = \{i_{t,1}, \dots, i_{t,n}\}$ is given online, where $i_{t,j} \in \mathbb{R}$ denotes the value of attribute a_j at current time t . With each \vec{i}_t given, a decision needs to be made instantly whether or not \vec{i}_t is anomalous.

Past data H (assumed to be nominal) is also accessible. H is an $m \times n$ matrix where the columns denotes the n monitored attributes and the rows maintain the values of these attributes over m time steps. H can be recorded from a complete operation of the UV that is known to be nominal (e.g., a flight with no known failures), or it can be created from the last m inputs that were given online, that is, $H = \{\vec{i}_{t-m-1}, \dots, \vec{i}_{t-1}\}$.

We demonstrate the problem using a running example. Consider a UAV with its actuators that collects and monitors n attributes, such as: air-speed, heading, altitude, roll pitch and yaw, and other telemetry and sensors data. The actuators provides input in a given frequency (usually with 10Hz frequency), when suddenly a fault occurs; for instance, the altimeter is stuck on a valid value, while the GPS's indicated that the altitude keeps on rising. Another example could be that the UAV's stick is moved left or right but the UAV is not responsive, due to icy wings. This is expressed in the unchanging values of the roll and heading. Our goal is to detect these failures, by flagging them as anomalies.

3.2 Online Detection

We utilize a *sliding window* technique [4] to maintain H , the data history, online. The sliding window (see Figure 1) is a dynamic window of predefined size m which governs the size of history taken into account in our algorithm. Thus, every time a new input \vec{i}_t is received, H is updated as $H \leftarrow \{\vec{i}_{t-m-1}, \dots, \vec{i}_{t-1}\}$ the last m online inputs. The data in H is always assumed to be nominal and is used in the *online training* process. Based on H we evaluate the anomaly score for the current input \vec{i}_t using the *Mahalanobis Distance* [13].

Mahalanobis Distance is an n dimensional Z -score. It calculates the distance between an n dimensional point to a group of others, in units of standard deviations [13]. In contrast to the common n dimensional Euclidean Distance, Mahalanobis Distance also considers the points' distribution. Therefore, if the group of points represents an observation, then the Mahalanobis Distance indicates whether a new point is an outlier compared to the observation. A point with similar values to the observed points is located in the multidimensional space, within a dense area and will have a lower Mahalanobis Distance. However, an outlier will be located outside the dense area and will have a larger Mahalanobis Distance.

An example is depicted in Figure 2. We can see in the figure that while A and B have the same Euclidean distance from the centroid μ , A 's Mahalanobis Distance (3.68) is greater than B 's (1.5), because an instance of B is more probable than an instance of A with respect to the other points.

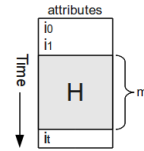


Figure 1: Illustration of the sliding window.

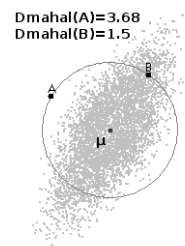


Figure 2: Euclidean vs. Mahalanobis Distance.

Thanks to the nature of the Mahalanobis Distance, we can utilize it for anomaly detection in our environment. Each of the n attributes of the domain correlates to a dimension. An input vector \vec{i}_t is the n dimensional point, that is measured by Mahalanobis Distance against H . The Mahalanobis Distance is then used to indicate whether each new input point \vec{i}_t is an outlier with respect to H .

Using the Mahalanobis Distance, we can easily detect the three common categories of anomalies [4]:

1. *Point anomalies*: illegal data instances, corresponding to illegal values in \vec{i}_t .
2. *Contextual anomalies*, that is, data instances that are only illegal with respect to specific context but not otherwise. In our approach, the context is provided by the changing data of the sliding window.
3. *Collective anomalies*, which are related data instances that are legal apart, but illegal when they occur together. This is met with the multi-dimensionality of the points being measured by the Mahalanobis Distance .

An anomaly of any type, can cause the representative point to be apart from the nominal points, in the relating dimension, thus placing it outside of a dense area, and leading to a large Mahalanobis Distance and eventually raising an alarm.

Formally, the Mahalanobis Distance is calculated as follows. Recall that $\vec{i}_t = (i_{t,1}, i_{t,2}, \dots, i_{t,n})$ is the vector of the current input of the n attributes being monitored, and $H = m \times n$ matrix is the group of these attributes' nominal values. We define the mean of H by $\mu = (\mu_1, \mu_2, \dots, \mu_n)$, and S is the covariance matrix of H . The Mahalanobis Distance, D_{mahal} , from \vec{i}_t to H is defined as:

$$D_{mahal}(\vec{i}_t, H) = \sqrt{(\vec{i}_t - \vec{\mu})S^{-1}(\vec{i}_t - \vec{\mu})^T}$$

Using the Mahalanobis Distance as an anomaly detector is prone to errors without guidance. Recently, Lin *et al.* [12] showed that the success of Mahalanobis Distance as an anomaly detector depends on whether the dimensions inspected are correlated or not. When the dimensions are indeed correlated, a larger Mahalanobis Distance can better indicate *point*, *contextual* or *collective* anomalies. However, the same effect occurs when uncorrelated dimensions are selected. When the dimensions are not correlated, it is more probable that a given nominal input point will differ from the observed nominal points in those dimensions, exactly as in contextual anomaly. This can cause the return of large Mahalanobis Distance and the generating of false alarms.

Therefore, it is imperative to use a *training process* prior to the usage of the Mahalanobis Distance. This process will *find* and *group* correlated attributes, after which Mahalanobis Distance can be *applied per each correlated set of attributes*. Instead of regarding \vec{i}_t as one n dimensional point and use one measurement of Mahalanobis Distance against H , we apply several measurements, one per each correlated set. In the next subsection we describe the work of the training process and how it is applied online.

3.3 Online Training

Finding correlated attributes automatically is a difficult task. Some attributes may be constantly correlated to more than one attribute, while other attribute's values can be dynamically correlated to other attributes based on the characteristics of the data. For example, the *elevation* value of an aircraft's stick is correlated to the aircraft's *pitch* and to the change of height, measured in the differences of the values of the *altitude* attribute. However, this is only true depending on the value of the *roll* attribute, which is influenced

by the *aileron* value of the aircraft's stick. As the aircraft is being rolled, the *pitch* axis is getting more vertical. This, in turn, makes the *elevation* value to correlate to the *heading* value, rather than the height. This example demonstrates how correlation between attributes can change during execution time. Thus, it is apparent that an *online* training is needed to find dynamic correlations between the attributes.

Figure 3 shows a visualization of a correlation matrix, where each cell $_{i,j}$ depicts the correlation strength between attributes a_i, a_j . The stronger the correlation, the darker the color of the cell. Figure 3 displays three snapshots taken from different time periods of a simulated flight, where 71 attributes were monitored. The correlation change is apparent.

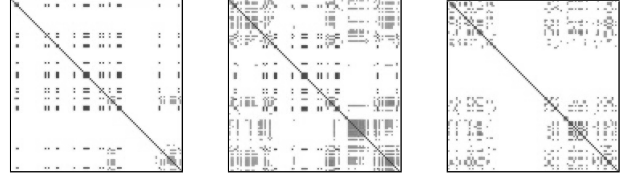


Figure 3: Visualization of correlation change during a flight

We use a fast online trainer, denoted as `Online_Trainer(H)`. Based on the data of the sliding window H , the online trainer returns n sets of dynamically correlated attributes, denoted as $CS = \{CS_1, CS_2, \dots, CS_n\}$, and a threshold per each set, denoted as $TS = \{threshold_1, \dots, threshold_n\}$.

The online trainer executes two procedures. The first is a correlation detector (see Alg. 1) that is based on *Pearson correlation coefficient* calculation. Formally, the Pearson correlation coefficient ρ between given two vectors \vec{X} and \vec{Y} with averages \bar{x} and \bar{y} , is defined as:

$$\rho = \frac{\sum_i (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_i (x_i - \bar{x})^2 \sum_i (y_i - \bar{y})^2}} \quad (1)$$

ρ ranges between $[-1, 1]$, where 1 represents a strong positive correlation, and -1 represents a strong negative correlation. Values closer to 0 indicate no correlation.

Algorithm 1 Correlation_Detector(H)

```

for each  $a_i \in A$  do
   $CS_i \leftarrow \phi$ 
  for each  $a_j \in A$  do
    if  $|\rho_{i,j}(H_i^T, H_j^T)| > ct$  then
      add  $a_j$  to  $CS_i$ 
  add  $CS_i$  to  $CS$ 
return  $CS$ 

```

Algorithm 1 returns the n sets of correlated attributes, one per each attribute $a_i \in A$. Each CS_i contains the indices of the other attributes that are correlated to a_i . The calculation is done as follows. The vectors of the last m values of each two attributes a_i, a_j are extracted from H and denoted H_i^T, H_j^T . We then apply the *Pearson correlation* on them denoted as $\rho_{i,j}$. If the absolute result $|\rho_{i,j}|$ is larger than a correlation threshold parameter $ct \in \{0..1\}$, then the attributes are declared correlated and a_j is added to CS_i .

The ct parameter governs the size of the correlated attributes set. On the one hand, the higher it is, less attributes are deemed correlated, thereby decreasing the dimensions and the total amount of calculations. However, this might also prevent attributes from being deemed correlated and affect the flagging of anomalies. On the other hand, the smaller the ct more attributes are considered correlated, thereby increasing the dimensions, and also increasing

the likelihood of false positives, as less correlated attributes are selected.

The second procedure sets a threshold value per each correlated set. These thresholds are later used by the *Anomaly Detector* (see Alg. 2) to declare an anomaly if the anomaly score of a given input crossed a threshold value. Each $threshold_a \in TS$ is set to be the highest Mahalanobis Distance of points with dimensions relating the attributes in CS_a extracted from H . Since every point in H is considered nominal, then any higher Mahalanobis Distance indicates an anomaly.

3.4 Specializing Anomaly Detection for Robots

Monitoring in the domains of autonomous robots is unique and have special characteristics. The main difference emerges from the fact that we are required to monitor using the data obtained from sensors that are used in the control loop to affect the environment. In other words, the expectations to see changes in the environment are a function of the actions selected by the agent.

Therefore, it makes sense to monitor the change in the values measured by the sensors (which originates from the robot’s actions), rather than the absolute values. The raw readings of the sensors usually do not correspond directly to the agent’s actions. For example, an increase of *speed* should be correlated to the lose of *height* generated by the UAV’s action, rather than correlating a specific *speed* value with a specific *height* value. Formally, we use the difference between the last two samples of each attribute, denoted as $\Delta(\vec{i}_t) = \vec{i}_t - \vec{i}_{t-1}$.

To eliminate false positives caused by the uncertainty inherent in the sensors’ readings, and also to facilitate the reasoning about the relative values of attributes, we apply a smoothing function using a z -transform. This filter measures changes in terms of standard deviations (based on the sliding window) and normalizes all values to using the same standard deviation units. A Z -score is calculated for a value x and a vector \vec{x} using the vector’s mean value \bar{x} and its standard deviation σ_x , that is, $Z(x, \vec{x}) = \frac{x - \bar{x}}{\sigma_x}$.

We then transform each value $i_{t,j}$ to its Z -score based on the last m values extracted from the sliding window H (H_j^T). Formally, $Z_{raw}(\vec{i}_t) = \{Z(i_{t,1}, H_1^T), \dots, Z(i_{t,n}, H_n^T)\}$. We also define this transformation on the differential data as $Z_\Delta(\vec{i}_t) = Z_{raw}(\Delta(\vec{i}_t))$.

Two aspects emphasize the need to use filters. First, the live feed of data is noisy. Had we used only the last two samples, the noise could have significantly damaged the quality of the differential data. Second, the data feed is received with high frequency. When the frequency of the incoming data is greater than the speed of the change in an attribute, the differential values might equal zero. Therefore, a filter that slows the change in that data, and takes into account its continuity, must be applied. In our simulations we experimented with two types of filters that use the aforementioned Z -transformations, Z_{raw} and Z_Δ .

When an actuator is idle, its Z -values are all 0s, since each incoming raw value is the same as the last m raw values. However, as the actuator’s reading changes, the raw values become increasingly different from one another, increasing the actuator’s Z -values, up until the actuator is idle again (possibly on a different raw value). The last m raw values are filled again with constant values, lowering the actuator’s Z -values. This way, a change is modeled by a “ripple effect”, causing other attributes that correspond to the same changes, also to be affected by that effect.

Figure 4 illustrates the Z -transformation technique. The data is taken from a segment of a simulated flight. The figure presents values of attributes (Y Axis) through time (X axis). The *aileron* attribute stores the left and right movement of the UAV’s stick. These

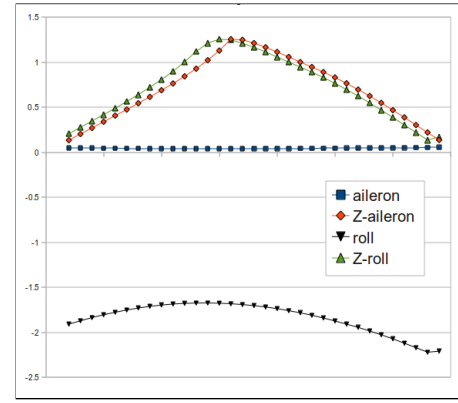


Figure 4: Illustration of the Z -transformation.

movements controls the UAV’s *roll* which is sensed using gyros and stored in the *roll* attribute. We say that the *aileron* and *roll* attributes are correlated if they share the same effect of change. The *aileron*’s raw data is shown in Figure 4 as the square points, which remains almost constant. Yet, the *roll*’s raw data, marked as an upside triangle, differs significantly from the *aileron*’s data. However, they share a similar ripple effect, illustrated by their Z -transformation values, shown in the triangle points and the diamond points. Thus, our *Pearson* calculation technique can find this correlation quite easily. Other attributes that otherwise could be mistakenly considered correlated when using just the raw data or Δ technique, will not be considered as such when using the Z -transformation technique, unless they both share a similar ripple effect. This could explain the fact that the Z_Δ technique was proven to be the best one that minimizes the number of false positives as described in Section 4.2

3.5 The Anomaly Detector

Algorithm 2 lists how the anomaly detector works. Each input vector that is obtained online, \vec{i}_t , is transformed to $Z_\Delta(\vec{i}_t)$. The sliding window H is updated. The *online trainer* process retrieves the *sets of correlated attributes* and their *thresholds*. For each correlated set, only the relating dimensions are considered when we compare the point extracted from \vec{i}_t to the points with the same dimensions in H . These points are compared using Mahalanobis Distance. If the distance is larger than the correlated sets’ threshold, then an anomaly is declared.

Algorithm 2 Anomaly_Detector(\vec{i}_t)

```

 $\vec{i}_t \leftarrow Z_\Delta(\vec{i}_t)$ 
 $H \leftarrow \{\vec{i}_{t-m-1}, \dots, \vec{i}_{t-1}\}$ 
 $CS, TS \leftarrow \text{Online\_Trainer}(H)$ 
for each  $a$  ( $0 \leq a \leq |CS|$ ) do
  Let  $CS_a$  be the  $a$ ’th set of correlated attributes in  $CS$ 
  Let  $threshold_a$  be the  $a$ ’th threshold, associated with  $CS_a$ 
   $P_H \leftarrow$  points with dimensions relating to  $CS_a$ ’s attributes extracted from  $H$ 
   $p_{new} \leftarrow$  point with dimensions relating to  $CS_a$ ’s attributes extracted from  $\vec{i}_t$ 
  if  $threshold_a < D_{mahal}(p_{new}, P_H)$  then
    declare “Anomaly”.

```

4. EVALUATION

First, we describe the experiments setup; the test domains and anomalies, the different anomaly detectors that emphasize that need of each of our approach’s features, and how the scoring is done. Then, we evaluate the influence of each feature of our ap-

proach, and we show how it outperforms other anomaly detection approaches.

4.1 Experiments Setup

We use three domains to test our approach, described in Table 1.

Domain	UAV	UGV	FlightGear
data	real	real	simulated
anomalies	simulated	real	simulated
scenarios	2	2	15
scenario duration (sec)	2100	96	660
attributes	55	25	23
frequency	4Hz	10Hz	4Hz
anomalies per scenario	1	1	4 to 6
anomaly duration (sec)	64, 100	30	35

Table 1: Tested domains and their characteristics.

The first is a commercial *UAV* (Unmanned Aerial Vehicles). The data of two real flights, with simulated faults, was provided by the manufacture. The fault of the first flight is a gradually decreasing value of one attribute. The fault of the second flight is an attribute that froze on a legal value. This fault is specially challenging, because it is associated with an attribute that is not correlated to any others, making it very difficult for our approach to detect the anomaly.

The second domain is a *UGV*. We used a laboratory robot, the *RV400* (see Fig. 5). This robot is equipped with ten sonars, four bumpers and odometry measures. We tested two scenarios. In each scenario the robot went straight, yet it was tangled with a string that was connected to a cart with weight. The extra weight causes the robot to slow down in the first scenario, and completely stop in the second scenario. These scenarios demonstrate anomalies that are a result of the physical objects which are not sensed by the robot. Therefore, the robot’s operating program is unaware of these objects as well, leaving the situation unhandled. This domain also presents the challenge of having little data (only 96 seconds of data).

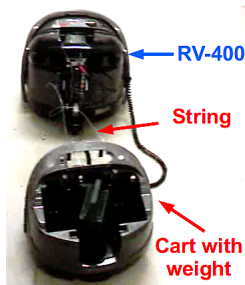


Figure 5: RV-400 tangled with a string connected to a heavy cart.



Figure 6: FlightGear flight simulator.

To further test our approach, on more types of faults and on various conditions, we used a third domain, the *FlightGear* flight simulator (see Fig. 6). *FlightGear* models real world behavior, and provides realistic noisy data. “Instruments that lag in real life, lag correctly in *FlightGear*, gyro drift is modeled correctly, the magnetic compass is subject to aircraft body forces.”[6] Furthermore, *FlightGear* also accurately models many instrument and system faults, that can be injected into a flight. For example, “if the vacuum system fails, the HSI gyros spin down slowly with a corresponding degradation in response as well as a slowly increasing bias/error.”[6]

In the *FlightGear* simulation, we programmed an autonomous UAV to fly according to the following behaviors: a take-off, an altitude maintenance, a turn, and eventually a landing. During a flight, 4 to 6 faults were injected into three different components; the *airspeed-indicator*, *altimeter* and the *magnetic compass*. The

faults and their time of injection, were both randomly selected. Each fault could be a contextual anomaly [4] with respect to the UAV’s behavior, and a collective anomaly [4] with respect to the measurements of different instruments such as the *GPS airspeed*, *altitude indicators* and the *Horizontal Situation Indicator*.

Our approach is based on three key features, compared to previous work. 1) a comparison to a *sliding window*, rather than a complete record of past data. 2) the use of an *online training process* to find correlated attributes. 3) the use of *differential filtered data*. To show the independent contribution of each feature we tested the following online anomaly detectors that are described by three parameters (*Nominal Data*, *Training*, *Filter*), as summarized in Table 2. The bold line is our recommended approach when using Z_{Δ} as the *filter*.

Name	Nominal Data	Training
(CD,none, <i>filter</i>)	complete past data	none
(SW,none, <i>filter</i>)	sliding window	none
(CD,Tcd, <i>filter</i>)	complete past data	offline
(SW,Tcd, <i>filter</i>)	sliding window	offline
(SW,Tsw, <i>filter</i>)	sliding window	online

Table 2: Tested Anomaly Detectors.

The *filter* can be *raw*, Δ , Z_{raw} , Z_{Δ} as described in Section 3.4. CD denotes the use of a Complete record of past Data. SW denotes the use of a Sliding Window. (SW,Tsw, Z_{Δ}) is our proposed anomaly detector described in section 3.5. (SW,Tcd,*filter*) uses almost the same technique; the thresholds are calculated on the data of the sliding window. However the training is done first, offline, on a complete record of past data. With (CD,Tcd,*filter*), the data of the sliding window is replaced with the data of the complete past record. With (SW,none,*filter*) no training is done, meaning all the dimensions are used at once to compare \vec{i}_t to the data of the sliding window. (CD,none,*filter*) uses all the dimensions to compare \vec{i}_t to the data of a complete past record.

(CD,Tsw,*filter*) is *not* displayed in table 2. This anomaly detector executes the training process on the sliding window, thus, thresholds are calculated online each time different correlated sets are returned. However, the comparison of the online input is made against a complete record of past data, thus, thresholds are calculated on the data of *CD*, which is considerably larger than the data of *SW*. Therefore, the anomaly detection of (CD,Tsw,*filter*) is not feasible online, hence, its is not compared to the other anomaly detectors displayed in table 2.

We evaluated the different anomaly detectors by the detection rate and false alarm rate. To this aim we define four counters, which are updated for every input \vec{i}_t . A “True Positive” (TP) refers to the flagging of an anomalous input as anomalous. A “False Negative” (FN) refers to the flagging of an anomalous input as nominal. A “False Positive” (FP) refers to the flagging of a nominal input as anomalous. A “True Negative” (TN) refers to the flagging of a nominal input as nominal. Table 3 summarizes how these counters are updated.

score	description
TP	counts 1 if at least one “anomalous” flagging occurred during a fault time
FN	counts 1 if no “anomalous” flagging occurred during a fault time
FP	counts every “anomalous” flagging during nominal time
TN	counts every “nominal” flagging during nominal time

Table 3: Scoring an anomaly detector.

For each algorithm, we calculated the detection rate = $\frac{tp}{tp+fn}$

and the false alarm rate = $\frac{fp}{fp+tn}$. An efficient classifier should maximize the detection rate and minimize the false alarm rate. The perfect classifier has a detection rate of 1, and a false alarm rate of 0.

4.2 Results

Figures 7 and 8 present the detection rate and the false alarm rate respectively of 15 flights in the *FlightGear* simulator. We present the influence of the different filters on the different algorithms. The scale ranges from 0 to 1, where 0 is the best possible score for a false alarm rate and 1 is the best possible score for a detection rate.

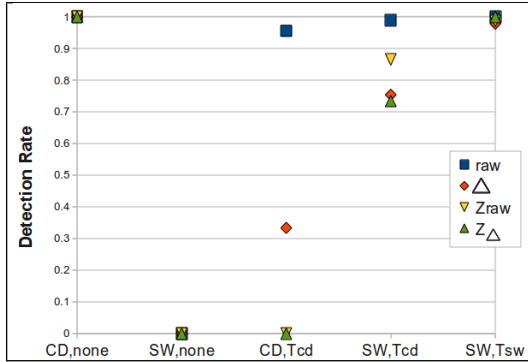


Figure 7: Detection rate. (Higher is better)

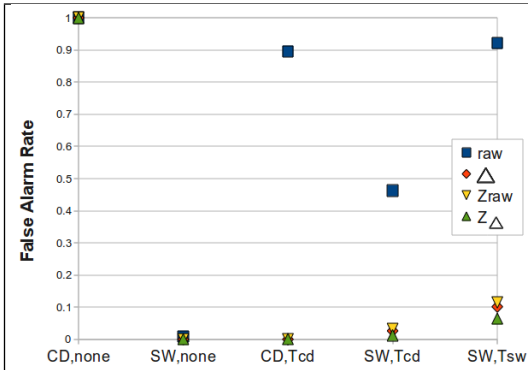


Figure 8: False alarm Rate. (Lower is better)

We begin with the first anomaly detector, (CD,none). Both Figures 7 and 8 show a value of 1, indicating a constant declaration of an anomaly. In this case, no improvement is achieved by any of the filters. This accounted for the fact that the comparison is made to a complete record of *past* data. Since the new point is sampled from a *different* flight, it is very unlikely for it be observed in the past data, resulting with a higher Mahalanobis Distance than the threshold, and the declaration of an anomaly.

The next anomaly detector we examine is (SW,none). In this detector, the comparison is made to the sliding window. Since data is collected in a high frequency, the values of i_t and the values of each vector in H , are very similar. Therefore the Mahalanobis Distance of i_t is not very different than the Mahalanobis Distance of any vector in H . Thus the threshold is very rarely crossed. This explains the very low false alarm rate for this algorithm in Figure 8. However, the threshold is not crossed even when anomalies occur, resulting in a very low detection rate as Figure 7 shows. The reason is the absence of training. The Mahalanobis Distance of a contextual or collective anomaly, is not higher than Mahalanobis Distances of points with uncorrelated dimensions in H . The anomalies are not conspicuous enough.

The next two anomaly detectors, introduce the use of offline training. The first (CD,Tcd), uses a complete record of past data, while the second (SW,Tcd) uses a sliding window. However in both anomaly detectors the training is done offline, on a complete record of past data. When no filter is used, (CD,Tcd) declares an anomaly most of the times, this is illustrated in the square dot in Figures 7 and 8. When filters are used, more false negatives occur, expressed in the almost 0 false alarm rates and the decreasing of the detection rate. However, when a sliding windows is used, even with no filters, (SW,Tcd) got better results, a detection rate of 1, and less than 0.5 false alarm rate, which is lower than (CD,Tcd)'s false alarm rate. The filters used with (SW,Tcd) lower the false alarm rate to almost 0, but this time, the detection rate, though decreased, remains high. Comparing (SW,Tcd) to (CD,Tcd) shows the importance of a sliding window, while comparing (SW,Tcd) to (SW,none) it shows the crucial need of training.

The final anomaly detector is (SW,Tsw) which differs from (SW,Tcd) by the training mechanism. (SW,Tsw) applies an online training on the sliding window. This allows achieving a very high detection rate. Each filter used allows increasing the detection rate closer to 1, until Z_{Δ} gets the score of 1. The false alarm rate is very high when no filter is used. When using filters we are able to reduce the false alarm rate to nearly 0. (SW,Tsw, Z_{Δ}), which is the approach we described in section 3.5, achieves a detection rate of 1, and a low false alarm rate of 0.064.

The results show the main contributions of each feature, summarized in table 4

feature	contribution	reason
sliding window	decreases FP	similarity of i_t to H .
training	increases TP	correlated dimensions \rightarrow more conspicuous anomalies.
online training	increases TP	correspondence to dynamic correlation changes.
filters	decreases FP increases TP	better correlations are found.

Table 4: Feature Contributions

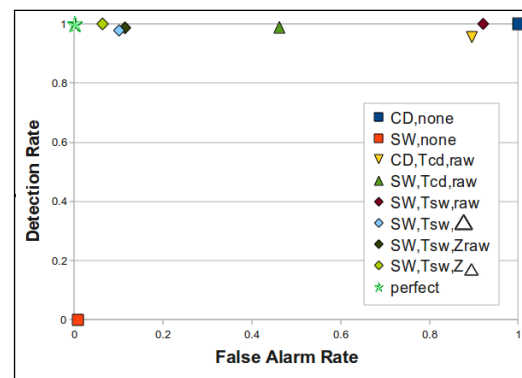


Figure 9: The classifier plane.

Figure 9 describes the entire space of classifiers: the X -axis is the false alarm rate and the Y -axis is the detection rate. A classifier is expressed as a $2D$ point. The perfect anomaly detector is located at point (0,1), that is, it has no false positives, and detects all the anomalies. Figure 9 illustrates that when the features of our approach are applied, they allow the results to approximate the perfect classifier.

Figure 10 shows the detection rates and false alarm rates of (TW,Tsw, Z_{Δ}) in the classifier space, when we increase the correlation threshold $ct \in \{0..1\}$ in the online trainer described in

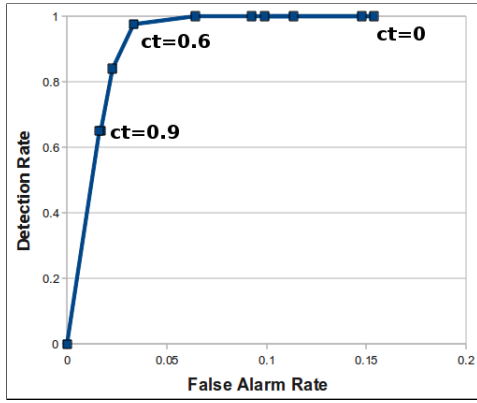


Figure 10: The influence of the correlation threshold.

section 3.3. Note that the X axis scales differently than in Figure 9, it ranges between $[0, 0.2]$ in order to zoom in on the effect.

When ct equals 0 all the attributes are selected for each correlated set, resulting with false alarms. As ct increases, less uncorrelated attributes are selected, reducing the false alarms, until a peak is reached. The average peak of the 15 *FlightGear*'s flights was reached when ct equals 0.5. (TW, Tsw, Z_{Δ}) averaged a detection rate of 1, and a false alarm rate of 0.064. As ct increases above that peak, less attributes that are crucial for the detection of an anomaly are selected, thereby increasing the false negatives, which in return lowers the detection rate. When ct reaches 1, no attributes are selected, resulting a constant false negative.

To further test our approach, we compare it with other methods.

Support Vector Machines (SVM) are considered very successful classifiers when examples of all categories are provided [17]. However, the SVM algorithm classifies every input as nominal, including all anomalies, resulting in a detection rate of 0 as Figure 11 shows. Samples of both categories are provided to the SVM, and it is an offline process, yet, the contextual and collective anomalies are undetected. This goes to show how illusive these anomalies are, which were undetected by a successful and well-known classifier, even under unrealistic favoring conditions.

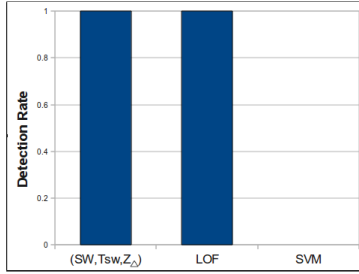


Figure 11: FlightGear Domain Detection Rate

We also examine the quality of (SW, Tsw, Z_{Δ}) in the context of other anomaly detectors. We compared it to the *incremental LOF algorithm* [15]. As in our approach, the *incremental LOF* returns a density based anomaly score in an online fashion. The *incremental LOF* uses K nearest neighbor technique to compare the density of the input's "neighborhood" against the average density of the nominal observations [15]. Figure 11 shows a detection rate of 1 to (SW, Tsw, Z_{Δ}) and the *incremental LOF algorithm*, making it a better competitive approach to ours than the SVM.

Since the *incremental LOF* returns an anomaly score rather than an anomaly label, we compared the two approaches using an offline *optimizer algorithm* that gets the anomaly scores returned by an anomaly detector, and the anomaly times, and returns the optimal thresholds, which in retrospect, the anomaly detector would have labeled the anomalies, in a way that all anomalies would have been detected with a minimum of false positives.

Figures 12 to 15 show for every tested domain the false alarm rate of

1. (SW, Tsw, Z_{Δ})
2. optimized (SW, Tsw, Z_{Δ}) denoted as $OPT(SW, Tsw, Z_{\Delta})$
3. optimized *incremental LOF* denoted as $OPT(LOF)$

The results of the detection rate for these anomaly detectors is 1 in every tested domain, just like the perfect classifier; all anomalies are detected. Thus, the false alarm rate presented, also expresses the distance to the perfect classifier, where 0 is perfect.

The comparison between (SW, Tsw, Z_{Δ}) to $OPT(LOF)$ does not indicate which approach is better in anomaly detection, since the *incremental LOF* is optimized, meaning, the best *theoretical* results it can get are displayed. However the comparison between $OPT(SW, Tsw, Z_{\Delta})$ to $OPT(LOF)$ does indicate which approach is better, since both are optimized. The comparison between $OPT(SW, Tsw, Z_{\Delta})$ to (SW, Tsw, Z_{Δ}) indicates how better (SW, Tsw, Z_{Δ}) can theoretically get.

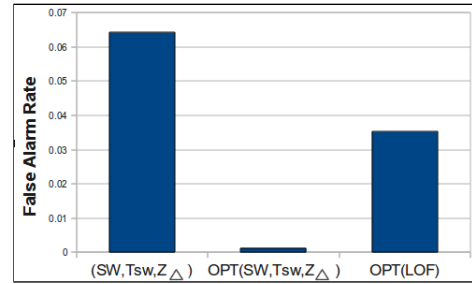


Figure 12: FlightGear domain.

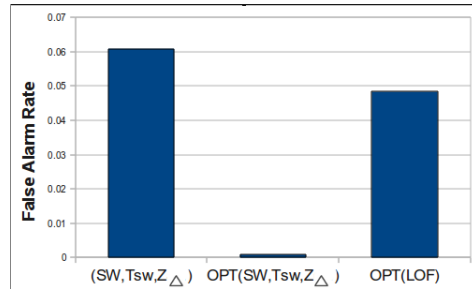


Figure 13: UAV first flight.

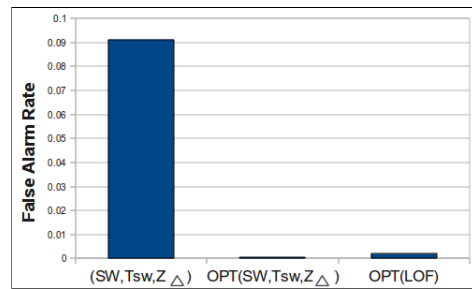


Figure 14: UAV second flight.

In all the domains the $OPT(SW, Tsw, Z_{\Delta})$ had the lowest false alarm rate. Naturally, $OPT(SW, Tsw, Z_{\Delta})$ has a lower false alarm rate than (SW, Tsw, Z_{Δ}) . But more significantly, it had a lower false alarm rate than $OPT(LOF)$, making our approach a better anomaly detector than the *incremental LOF algorithm*. Of all the tested domains, the highest false alarm rate of (SW, Tsw, Z_{Δ}) occurred in the UAV's second flight, as Figure 14 show (little above 0.09). In this flight, the fault occurred in an attribute that is not very correlated

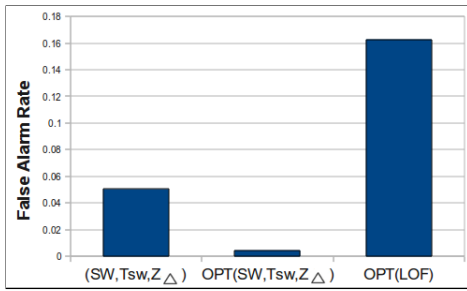


Figure 15: UGV domain.

to any other. Thus, the correlation threshold (ct) had to be lowered. This allowed the existence of a correlated set that includes the faulty attribute as well as other attributes. This led to the detection of the anomaly. However the addition of uncorrelated attributes increased the false alarm rate as well.

Figure 15 show a surprising result. Even though the results of the *incremental LOF* are optimized, (SW,Tsw,Z Δ), which is not optimized, had a lower false alarm rate. This is explained by the fact that in the UGV domain, there was very little data. KNN approaches usually fail when nominal or anomalous instances do not have enough close neighbors [4]. This domain simply did not provide the LOF calculation enough data to accurately detect anomalies. However, the Mahalanobis Distance uses all the points in the distribution, enough data to properly detect the anomalies.

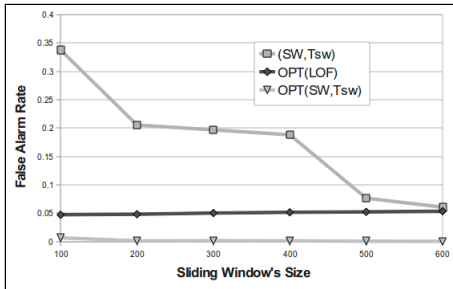


Figure 16: Sliding Window's changing size.

Figure 16 shows the false alarm rate influenced by the increase of the sliding window's size. While Mahalanobis Distance uses the distribution of all the points in the sliding window, the KNN uses only a neighborhood within the window, thus unaffected by its size. Therefore, there exists a size upon which our approach's *real* false alarm rate, meets the *incremental LOF's optimized* false alarm rate.

5. SUMMARY AND FUTURE WORK

We showed an unsupervised, model free, online anomaly detector for robots, that shows a great potential in detecting anomalies while minimizing false alarms. Moreover, the features of the sliding window, the online training and the filtered differential data, made the difference between having an unusable anomaly detector, and an anomaly detector that is better than current existing methods, when applied to robots. However we also showed that with different thresholds, even better results could be obtained. Therefore, in our future work we shall try to select thresholds in a more clever way. Raising an alarm is just the first step towards autonomous self-correcting robots. The next step before diagnosing the cause of the fault, is isolating it. By process of eliminating dimensions, the anomaly, or fault, could be isolated, thus helping a diagnosis process.

Acknowledgments. This research was supported in part by ISF grant #1357/07. As always, thanks to K. Ushi and K. Raviti.

6. REFERENCES

- [1] N. Agmon, S. Kraus, and G. A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *ICRA*, pages 2339–2345, 2008.
- [2] A. Birk and S. Carpin. Rescue robotics - a crucial milestone on the road to autonomous systems. *Advanced Robotics Journal*, 20(5), 2006.
- [3] T. Brotherton and R. Mackey. Anomaly detector fusion processing for advanced military aircraft. In *IEEE Proceedings on Aerospace Conference*, pages 3125–3137, 2001.
- [4] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):1–58, 2009.
- [5] L. Cork and R. Walker. Sensor fault detection for UAVs using a nonlinear dynamic model and the IMM-UKF algorithm. *IDC*, pages 230–235, 2007.
- [6] FlightGear. Website, 2010. <http://www.flightgear.org/introduction.html>.
- [7] FlightGear in Research. Website, 2010. <http://www.flightgear.org/Projects/>.
- [8] P. Goel, G. Dedeoglu, S. I. Roumeliotis, and G. S. Sukhatme. Fault-detection and identification in a mobile robot using multiple model estimation and neural network. In *ICRA*, 2000.
- [9] M. A. Goodrich, B. S. Morse, D. Gerhardt, J. L. Cooper, M. Quigley, J. A. Adams, and C. Humphrey. Supporting wilderness search and rescue using a camera-equipped mini UAV. *Journal of Field Robotics*, pages 89–110, 2008.
- [10] R. M. J. Craighead and B. G. J. Burke. A survey of commercial open source unmanned vehicle simulators. In *ICRA*, pages 852–857, 2007.
- [11] J. Laurikkala, M. Juhola, and E. Kentala. Informal identification of outliers in medical data. In *Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology*. 2000.
- [12] R. Lin, E. Khalastchi, and G. A. Kaminka. Detecting anomalies in unmanned vehicles using the mahalanobis distance. In *ICRA*, pages 3038–3044, 2010.
- [13] P. C. Mahalanobis. On the generalized distance in statistics. In *Proceedings of the National Institute of Science*, pages 49–55, 1936.
- [14] T. Oates, M. D. Schmill, D. E. Gregory, and P. R. Cohen. *Learning from Data: Artificial Intelligence and Statistics*, chapter Detecting Complex Dependencies in Categorical Data, pages 185–195. Springer Verlag, 1995.
- [15] D. Pokrajac. Incremental local outlier detection for data streams. In *IEEE Symposium on Computational Intelligence and Data Mining*, 2007.
- [16] E. F. Sorton and S. Hammaker. Simulated flight testing of an autonomous unmanned aerial vehicle using flight-gear. AIAA 2005-7083, Institute for Scientific Research, Fairmont, West Virginia, USA, 2005.
- [17] I. Steinwart and A. Christmann. *Support Vector Machines*. Springer-Verlag, 2008.
- [18] P. Sundvall and P. Jensfelt. Fault detection for mobile robots using redundant positioning systems. In *ICRA*, pages 3781–3786, 2006.
- [19] S. Thrun. Robotic mapping: A survey. In *Exploring Artificial Intelligence in the New Millenium*, pages 1–35. Morgan Kaufmann, 2003.

Tree Adaptive A^{*}

Carlos Hernández

Departamento de
Ingeniería Informática
Universidad Católica
de la Sma. Concepción
Caupolicán 491, Concepción, Chile
chernan@ucsc.cl

Xiaoxun Sun Sven Koenig

Computer Science
Department
University of
Southern California
Los Angeles, CA 90089, USA
{xiaoxuns,skoenig}@usc.edu

Pedro Meseguer

Institut d'Investigació en
Intel·ligència Artificial
IIIA-CSIC
Campus UAB
08193 Bellaterra, Spain
pedro@iiia.csic.es

ABSTRACT

Incremental heuristic search algorithms can solve sequences of similar search problems potentially faster than heuristic search algorithms that solve each search problem from scratch. So far, there existed incremental heuristic search algorithms (such as Adaptive A^{*}) that make the h-values of the current A^{*} search more informed, which can speed up future A^{*} searches, and incremental heuristic search algorithms (such as D^{*} Lite) that change the search tree of the current A^{*} search to the search tree of the next A^{*} search, which can be faster than constructing it from scratch. In this paper, we present Tree Adaptive A^{*}, which applies to goal-directed navigation in unknown terrain and builds on Adaptive A^{*} but combines both classes of incremental heuristic search algorithms in a novel way. We demonstrate experimentally that it can run faster than Adaptive A^{*}, Path Adaptive A^{*} and D^{*} Lite, the top incremental heuristic search algorithms in the context of goal-directed navigation in unknown grids.

Categories and Subject Descriptors

I.2.8 [Problem Solving, Control Methods, and Search]:
[Graph and tree search strategies]

General Terms

Algorithms, Experimentation

Keywords

Agent Reasoning::Planning (single and multi-agent), Robot Reasoning::Planning, Path Planning

*This material is based upon work supported by NSF (while Sven Koenig was serving at NSF). It is also based upon work supported by Fondecyt-Chile under contract/grant number 11080063, ARL/ARO under contract/grant number W911NF-08-1-0468, ONR in form of a MURI under contract/grant number N00014-09-1-1031, DOT under contract/grant number DTFH61-11-C-00010 and the Spanish Ministry of Science and Innovation under grant number TIN2009-13591-C02-02. 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.

Cite as: Tree Adaptive A^{*}, Carlos Hernández, Xiaoxun Sun, Sven Koenig and Pedro Meseguer, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 123-130.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

1. INTRODUCTION

Agents, such as robots and game characters, have to be able to navigate from their current location to a given destination [4]. However, they might not know a map of the terrain initially, and their sensors can typically sense the terrain only near their current location. They can use path planning with the freespace assumption to navigate from their current location to their destination, which is a popular approach in robotics [13]: The agents plan a minimum-cost path from their current location to their destination under the assumption that the terrain is traversable except for the obstacles that they have already sensed. As they move along the planned path, they sense additional obstacles and add them to their map. When they detect obstacles on the path, they replan a minimum-cost path from their current location to their destination and repeat the process until they reach their destination or can no longer find a path to their destination (in which case the destination is unreachable).

Path planning with the freespace assumption thus interleaves path planning with movement and requires repeated searches. These searches need to be fast since agents have to move smoothly and without delay. For example, the computer game company Bioware imposes a time limit of 1-3ms on each search [2]. However, even A^{*} searches [6] can be time consuming if the terrain is large or many agents perform simultaneous searches. Incremental heuristic search algorithms use information from the current and previous searches to solve future similar search problems potentially faster than heuristic search algorithms that solve each search problem from scratch [12]. They have been used to speed up A^{*} searches in the context of both symbolic planning [9] and path planning [12]. There are two classes of incremental heuristic search algorithms:

- Incremental heuristic search algorithms of the first class make the h-values of the current search more informed, which can speed up future searches by making them more focused. Examples include Adaptive A^{*} [11], Generalized Adaptive A^{*} [19] and Multi-target Adaptive A^{*} [16].
- Incremental heuristic search algorithms of the second class change the search tree of the current search to the search tree of the next search, which can be faster than constructing it from scratch. Examples include D^{*} [18] and D^{*} Lite [10], which can speed up A^{*} searches by more than one order of magnitude [10] and are typically faster than Adaptive A^{*} and Generalized Adaptive A^{*} in the context of goal-directed navigation in unknown terrain [11]. Versions of them have been used as part of path planners in

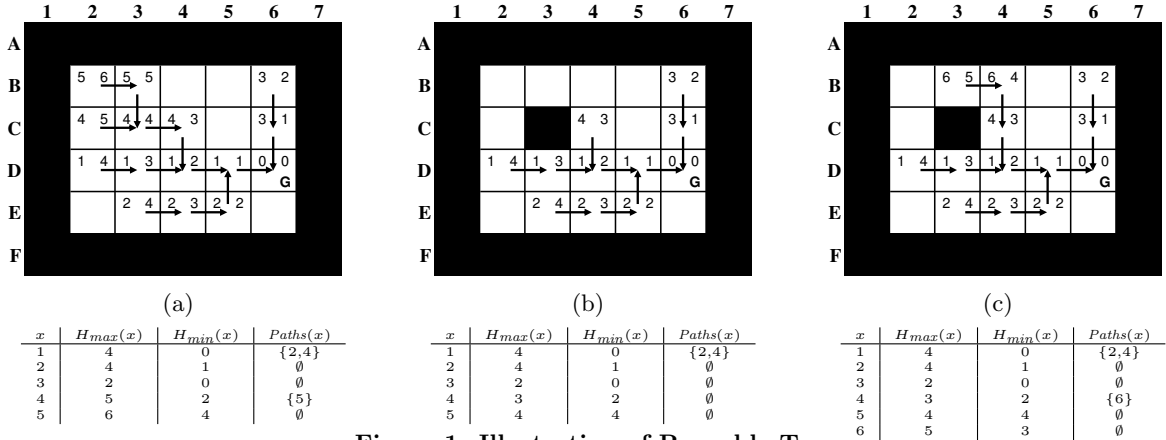


Figure 1: Illustration of Reusable Trees

a wide range of fielded robotics systems [14, 5, 15], including the winning DARPA Urban Challenge entry from Carnegie Mellon University.

We present Tree Adaptive A* (Tree-AA*) in this paper, an incremental heuristic search algorithm that applies to path-planning with the freespace assumption for goal-directed navigation in unknown terrain or, more generally, repeatedly following a minimum-cost path from the current location to a destination where the movement costs can increase (but not decrease). Tree-AA* generalizes Path Adaptive A* (Path-AA*) [7]. Path-AA* applies to path planning with the freespace assumption and generalizes Adaptive A* to reuse a suffix of the minimum-cost path of the current A* search (= reusable path) to allow the next A* search to terminate earlier. Tree-AA* also applies to path planning with the freespace assumption but generalizes Adaptive A* to reuse suffixes of the minimum-cost paths of the current and all previous A* searches (= reusable tree). Thus, Tree-AA* combines incremental heuristic search algorithms of the two above classes in a novel way. The reusable tree of Tree-AA* is similar to the search tree of incremental heuristic search algorithms (such as D* Lite) that change the search tree of the current A* search to the search tree of the next A* search since they perform backward A* searches to guarantee that the root of the search tree does not change. However, Tree-AA* changes the reusable tree via forward A* searches, which is a novel way of maintaining the search tree. We demonstrate experimentally that it can run faster than Path-AA*, Adaptive A* and D* Lite, the top incremental heuristic search algorithms in the context of goal-directed navigation in unknown grids.

2. NOTATION

We use the following notation: S is the finite set of states, which correspond to the locations. $s_{start} \in S$ is the current state of the agent, which corresponds to its current location. $s_{goal} \in S$ is the goal state, which corresponds to its destination. $Succ(s) \subseteq S$ is the set of successor states of state $s \in S$. $c(s, s') > 0$ is the cost of moving from state $s \in S$ to its successor state $s' \in Succ(s)$. The goal cost of a state is the cost of a minimum-cost path from the state to the goal state. The h-value $h(s)$ (= heuristic) of state $s \in S$ is a consistent approximation of the goal cost of the state, that is, one that satisfies the triangle inequality [17].

3. BACKGROUND

We provide a brief introduction to Adaptive A* and Path-AA* since Tree-AA* uses their principles. Both incremental heuristic search algorithms apply to path planning with the freespace assumption and use A* searches to find a minimum-cost path from the current state of the agent to the goal state. They perform A* searches from the current state of the agent to the goal state (= forward search), which is the most efficient search direction for Adaptive A* [10] and the only possible search direction for Path-AA*. As the agent follows the planned path, it senses additional obstacles, which increase the costs of moving from some states to their successor states (often to infinity). When one or more edges with increased costs are on the planned path between the current state of the agent and the goal state, Path-AA* replans a minimum-cost path from the current state of the agent to the goal state and then repeats the process until it reaches the goal state or it can no longer find a path to the goal state.

3.1 Adaptive A*

Adaptive A* [11] is based on the following “update principle,” which was first described in [8] in the context of hierarchical A* search: If the h-value of every state expanded by an A* search with consistent h-values is set to the f-value of the goal state minus the g-value of the state, then the resulting h-values are again consistent and weakly dominate the original h-values. Thus, an A* search with the resulting h-values expands no more states than an A* search with the original h-values (and the same tie-breaking strategy). The goal state has to remain unchanged from A* search to A* search but the start state can change and some movement costs can increase (but not decrease). Thus, Adaptive A* can be used for path planning with the freespace assumption, which typically makes the A* searches more focused and thus speeds them up. The properties of Adaptive A* are explained in more detail in [11]. We make extensive use of the property that Adaptive A* sets the h-values of all states on the minimum-cost path to their goal costs.

3.2 Path Adaptive A* (Path-AA*)

Path Adaptive A* (Path-AA*) [7] is based on the following “termination principle” and extends the “path-caching strategy,” which was first described in [8] in the context of hierarchical A* search: If one knows a minimum-cost path

from some state to the goal state (= reusable path) and the h-values of all states on the reusable path are equal to their goal costs, then a forward A* can terminate when it is about to expand a state on the reusable path (including the goal state). Thus, a Path-AA* search can terminate earlier than a regular A* search, that terminates only when it is about to expand the goal state. The (minimum-cost) path from the current state of the agent to the state on the reusable path and the (minimum-cost) path from the state on the reusable path to the goal state along the reusable path then form a minimum-cost path from the current state of the agent to the goal state. The properties of Path-AA* are explained in more detail in [7].

4. TREE ADAPTIVE A* (TREE-AA*)

Path-AA* was the first search algorithm to combine incremental heuristic search algorithms of the two above classes in a novel way. It is often faster than D* Lite, an alternative state-of-the-art incremental heuristic search algorithm [7], but has an important limitation: Path-AA* reuses only one path for the next A* search, namely a suffix of the minimum-cost path of the current A* search (= reusable path). In complex terrain, including terrain with large obstacles, the next A* search is unlikely to expand a state on that path far away from the goal state and thus unlikely to terminate much earlier than a regular A* search. We introduce Tree Adaptive A* (Tree-AA*) to address this limitation. Tree-AA* generalizes Path-AA* to reuse suffixes of the minimum-cost paths of the current and all previous A* searches (= reusable tree). It maintains minimum-cost paths from several states to the goal state organized in form of a tree rooted in the goal state. If one knows minimum-cost paths from several states to the goal state (= reusable tree) and the h-values of all states in the reusable tree are equal to their goal costs, then a forward A* search can terminate when it is about to expand a state in the reusable tree (including the goal state), for the same reasons as in the context of Path-AA*. Tree-AA* needs to support two operations, namely adding a path to the reusable tree and removing paths from the reusable tree:

- **Adding a Path to the Reusable Tree:** When an A* search of Tree-AA* terminates because it is about to expand a state in the reusable tree (including the goal state), then Tree-AA* adds the path from the current state of the agent to the state in the reusable tree to the reusable tree. It does this because the (minimum-cost) path from the current state of the agent to the state in the reusable tree and the (minimum-cost) path from the state in the reusable tree to the goal state along the branch of the reusable tree form a minimum-cost path from the current state of the agent to the goal state (since Adaptive A* finds minimum-cost paths) and the h-values of all states on the path are equal to their goal costs (since Adaptive A* updates the h-values this way).
- **Removing Paths from the Reusable Tree:** When the costs of edges in the reusable tree increase, then Tree-AA* uses the largest prefix of the reusable tree that does not contain edges with increased costs. (By prefix of a tree we mean the top part of the tree that includes its root.) It does this because all branches of the resulting tree are minimum-cost paths from some state to the goal state and the h-values of all states in the resulting tree are

still equal to their goal costs. When the cost of an edge from state s to state s' increases, then Tree-AA* finds the largest prefix of the reusable tree by removing both the edge and the subtree rooted in state s from the reusable tree.

4.1 Implementation of the Reusable Tree

Tree-AA* implements the above two operations efficiently by maintaining two variables for every state and three variables for every path $x = s_0 \dots s_n$ in the reusable tree, where s_0 is the state at the start of the path and s_n is the state at the end of the path that an A* search was about to expand when it terminated. We say that the states $s_0 \dots s_{n-1}$ belong to path x . Every path in the reusable tree is identified with a unique integer that corresponds to the number of the A* search after which it was added to the reusable tree (starting with one). Every path in the reusable tree is the prefix of a minimum-cost path from some state to the goal state. The h-values of all states on the path are equal to their goal costs and thus are strictly monotonically decreasing along the path. The variables are as follows:

- $Id(s)$ is the path in the reusable tree which state s belongs to. These values are initialized to zero, which means that state s is either the goal state or not in the reusable tree.
- $Reusabletree(s)$ is the parent of state s in the reusable tree if state s is a non-goal state in the reusable tree.
- $H_{max}(x)$ is the largest h-value of any state $s_0 \dots s_n$, that is, $H_{max}(x) = h(s_0)$. $H_{max}(0) = -1$, as explained below.
- $H_{min}(x)$ is the smallest h-value of any state $s_0 \dots s_n$, that is, $H_{min}(x) = h(s_n)$.
- $Paths(x)$ is the set of all paths in the reusable tree that connect to one of the states $s_0 \dots s_{n-1}$. These paths “feed into” path x .

Figure 1(a) shows a fictitious example of a reusable tree. The terrain is discretized into cells that are either blocked or unblocked, a common practice in the context of real-time computer games [1]. We assume for simplicity that the agent can move in the four main compass directions with cost one and thus operates on undirected four-neighbor grids. Values are shown only for cells that are in the reusable tree. A cell is black if it is blocked and this fact is known to the agent. The Id -value of a cell is shown in its upper left corner. The h-value of a cell is shown in its upper right corner. The $Reusabletree$ -pointer of a cell is shown as an arrow. The arrows thus show the reusable tree, that consists of five paths. Path 1 is D2 D3 D4 D5 D6, path 2 is E3 E4 E5 D5, path 3 is B6 C6 D6, path 4 is C2 C3 C4 D4, and path 5 is B2 B3 C3. The table shows the variables of all paths.

4.2 Implementation of the Operations

The goal state is always in the reusable tree. Tree-AA* could check whether $Id(s) > 0$ when it needs to check whether a non-goal state s is in the reusable tree. However, this would require it to set $Id(s)$ to zero when it removes a non-goal state from the reusable tree, which is expensive since Tree-AA* often needs to remove whole paths from the reusable tree. Thus, Tree-AA* checks whether $h(s) \leq H_{max}(Id(s))$ when it needs to check whether a non-goal state s is in the reusable tree. (The goal state fails this test.)

Tree-AA* can now remove a path x from the reusable tree by setting $H_{max}(x)$ to $H_{min}(x)$ without having to set $Id(s)$ to zero for all states s that belong to path x . Thus, $Id(s) = 0$ is not necessarily true for states s not in the reusable tree. There are two subtleties here. Non-goal states s that have not yet been part of the reusable tree correctly fail the test since $H_{max}(Id(s)) = H_{max}(0) = -1$. Non-goal states s that were part of the reusable search tree but have subsequently been removed correctly fail the test since they have an h-value larger than $H_{max}(Id(s))$. Tree-AA* adds a path to the reusable tree and removes paths from the reusable tree as follows:

- **Adding a Path to the Reusable Tree:** Tree-AA* adds a path $x = s_0 \dots s_n$ to the reusable tree as follows. It equates x with the number of the current A* search (as given by the variable *Counter*) to identify the path with a unique integer. It inserts x into the set $Paths(Id(s_n))$ if s_n is a non-goal state since path x feeds into the path that state s_n belongs to [Lines 9-10]. (The line numbers refer to the pseudo code of Tree-AA* in Figure 2). It sets $H_{min}(x)$ to $h(s_n)$ [Line 11] and $H_{max}(x)$ to $h(s_0)$ [Line 12] since the h-values are strictly monotonically decreasing along the path. It sets $Paths(x)$ to the empty set [Line 13] since no paths feed into path x yet. It sets $Id(s)$ to x and $Reusabletree(s)$ to the successor of state s on path x for all states $s_0 \dots s_{n-1}$ [Lines 14-18] since the states $s_0 \dots s_{n-1}$ belong to path x . The runtime of adding a path to the reusable tree is thus basically proportional to the number of states on the path.

- **Removing Paths from the Reusable Tree:** When the cost of an edge from state s to state s' increases, then Tree-AA* removes paths from the reusable tree as follows. If $Reusabletree(s) = s'$ then the edge might be in the reusable tree [Lines 69-70], namely on path $x := Id(s)$ [Line 20]. In this case, Tree-AA* sets $H_{max}(x)$ to $h(s')$ (if it was larger) to shorten path x [Lines 21-22]. It also removes all paths $x' \in Paths(x)$ with $H_{max}(x) \leq H_{min}(x')$ from the set $Paths(x)$ and schedules them for removal from the reusable tree [Lines 24-27]. For each path x scheduled for removal with $H_{max}(x) > H_{min}(x)$, it sets $H_{max}(x)$ to $H_{min}(x)$, removes all paths $x' \in Paths(x)$ from the set $Paths(x)$ and schedules them recursively for removal [Lines 28-34]. The runtime of removing paths from the reusable tree when the cost of one edge increases is thus basically proportional to the number of paths in the reusable tree, which is bounded by the number of A* searches performed so far.

Figure 1(b) continues the fictitious example from Figure 1(a) by showing the reusable tree after Tree-AA* removed paths from the reusable tree after C3 became blocked. Tree-AA* shortened path 4 to C4 D4 and removed path 5. Figure 1(c) shows the reusable tree after Tree-AA* added B3 B4 C4 D4 to the reusable tree after an A* search.

4.3 Pseudocode

We now put all of our insights together. The pseudo code of Tree-AA* in Figure 3 proceeds as follows:¹ It sets $Id(s)$ to

¹Tree-AA* maintains the following variables for its regular A* searches: *Counter* is the number of the current A* search. *OPEN* is the open list of the current A* search. *CLOSED* is the closed list of the current A* search. *Generated(s)* is the number of the

```

01 procedure InitializeState(s)
02 if (Generated(s) = 0)
03   g(s) := ∞;
04   h(s) := H(s);
05 else if (Generated(s) ≠ Counter)
06   g(s) := ∞;
07   Generated(s) := Counter;
08 procedure AddPath(s)
09 if (s ≠ sgoal)
10   insert Counter into Paths(Id(s));
11   Hmin(Counter) := h(s);
12   Hmax(Counter) := h(sstart);
13   Paths(Counter) := ∅;
14   while (s ≠ sstart)
15     saux := s;
16     s := Searchtree(s);
17     Id(s) := Counter;
18     Reusable(s) := saux;
19 procedure RemovePaths(s)
20 x := Id(s);
21 if (Hmax(x) > h(Reusable(s)))
22   Hmax(x) := h(Reusable(s));
23   QUEUE := ∅;
24   for all x' ∈ Paths(x)
25     if (Hmax(x) < Hmin(x'))
26       add x' to the end of QUEUE;
27       remove x' from Paths(x);
28   while QUEUE ≠ ∅
29     remove x from the head of QUEUE;
30     if (Hmax(x) > Hmin(x))
31       Hmax(x) := Hmin(x);
32       for all x' ∈ Paths(x)
33         add x' to the end of QUEUE;
34         remove x' from Paths(x);
35 function ComputePath()
36 while (OPEN ≠ ∅)
37   remove state s with the smallest g(s) + h(s) value from OPEN;
38   if (s = sgoal OR h(s) ≤ Hmax(Id(s)))
39     /* s is in reusable tree */
40     for all s' ∈ CLOSED
41       h(s') := g(s) + h(s) - g(s');
42     AddPath(s);
43     return true;
44   insert s into CLOSED;
45   for all s' ∈ Succ(s)
46     InitializeState(s');
47     if (g(s') > g(s) + c(s, s'))
48       g(s') := g(s) + c(s, s');
49     Searchtree(s') := s;
50     if (s' ∈ OPEN)
51       remove s' from OPEN;
52     insert s' into OPEN with value g(s') + h(s');
53 return false;
54 function Main()
55 Counter := 1;
56 Hmax(0) := -1;
57 for all s ∈ S
58   Generated(s) := Id(s) := 0;
59   Reusable(s) := NULL;
60 while (sstart ≠ sgoal)
61   InitializeState(sstart);
62   g(sstart) := 0;
63   OPEN := CLOSED := ∅;
64   insert sstart into OPEN with value g(sstart) + h(sstart);
65   if (ComputePath() = false)
66     return false; /* failure: the goal state is unreachable */
67   while (h(sstart) ≤ Hmax(Id(sstart)))
68     /* sstart is non-goal state in reusable tree */
69     sstart := Reusable(sstart);
70     for all increased costs c(s, s')
71       if (Reusable(s) = s')
72         RemovePaths(s);
73   Counter := Counter + 1;
74 return true; /* success: the goal state has been reached */

```

Figure 2: Tree-AA*

last A* search that generated state s . A* uses these values to initialize the g-values and h-values of states as needed during an A* search [procedure InitializeState] to avoid having to initialize them for all states before every A* search. *Searchtree(s)* is the parent of state s in the search tree of the *Generated(s)*th A* search. *g(s)* is the g-value of state s during the *Generated(s)*th A* search. *h(s)* is the current h-value of state s .

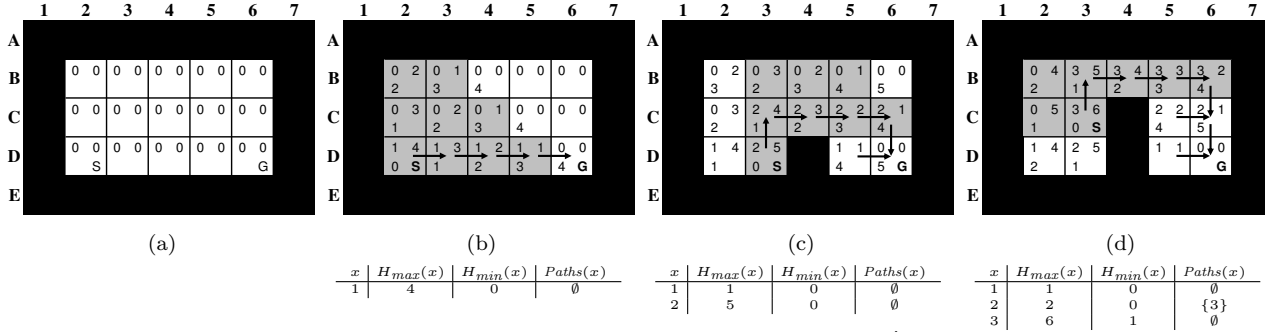


Figure 3: Example Trace of Tree-AA*

zero [Line 57], *ReusableTree(s)* to *NULL* [Line 58] and $h(s)$ (when needed for the first time during an A* search) to the user-given consistent h-value $H(s)$ [Line 4] for all states s . It performs a forward A* search [function *ComputePath*] until it is about to expand a state s in the reusable tree (including the goal state) [Line 38, termination principle]. It sets the h-value of every expanded state s' to the f-value of state s (which is the same as the f-value of the goal state) minus the g-value of state s' [Lines 39-40, update principle]. It then adds the (minimum-cost) path from the current state of the agent to state s to the reusable tree [procedure *AddPath*], as described above [Line 41]. It then follows the minimum-cost path from the current state of the agent to the goal state along the branch of the reusable tree [Lines 66-70]. Whenever edge costs increase, it removes paths from the reusable tree [procedure *RemovePaths*], as described above [Lines 68-70]. If the current state of the agent is no longer in the reusable tree, it performs another forward A* search and then repeats the process until the agent either reaches the goal state or it can no longer find a path to the goal state. The correctness proof of Tree-AA* is basically the same as that of Path-AA* and thus not given here.

4.4 Example Trace of Tree-AA*

Figure 3(a-d) shows the beginning of a trace of Tree-AA*. The agent always senses the blockage status of its four neighboring cells and can then move to any one of the unblocked neighboring cells with cost one. Its task is to move from start cell S to goal cell G . It assumes that all cells are unblocked except for the blocked cells that it has already sensed. It plans a minimum-cost path from its current cell to the goal cell. As it follows the planned path, it senses additional blocked cells and adds them to its map. When it detects blocked cells on its path, it replans a minimum-cost path from its current cell to the goal cell until it reaches the goal cell or can no longer find a path to the goal cell. The user-given h-values are all zero. A cell is black if it is blocked and this fact is known to the agent. The Id-value of a cell is shown in its upper left corner. The h-value of a cell (after it was updated using the update principle) is shown in its upper right corner. The g-value of a cell is shown in its lower left corner if it was generated during the current A* search. A cell is shaded if it was expanded during the current A* search. The ReusableTree-pointer of a cell is shown as an arrow if it belongs to the reusable tree.

Figure 3(a) shows the initial situation with start cell D2. Figure 3(b) shows that the first A* search of Tree-AA* from D2 to D6 terminates when it is about to expand D6 and returns the D2 D3 D4 D5 D6. Tree-AA* adds the path to

the reusable tree and updates the h-values of the expanded states using the update principle. The agent then follows the branch of the reusable tree from D2 to D3, where it senses that D4 is blocked. Tree-AA* removes D2 D3 D4 D5 from the reusable tree. Figure 3(c) shows that the second A* search of Tree-AA* from D3 to D6 terminates when it is about to expand D6 and returns D3 C3 C4 C5 C6 D6. It expands fewer cells than an A* search with the user-given zero h-values (which also expands B2, C2 and D2), illustrating the speed up achieved with the update principle. Tree-AA* adds the path to the reusable tree and updates the h-values. The agent then moves from D3 to C3, where it senses that C4 is blocked. Tree-AA* removes D3 C3 C4 C5 from the reusable tree. Figure 3(d) shows that the third A* search of Tree-AA* from C3 to D6 terminates when it is about to expand C6 and returns C3 B3 B4 B5 B6 C6. It terminates earlier than a regular A* search with the same h-values (which also expands C6 and terminates only when it is about to expand D6), illustrating the speed up achieved with the termination principle.

4.5 Comparison of Path-AA* and Tree-AA*

Figure 4 shows an example that illustrates the difference between Path-AA* (top) and Tree-AA* (bottom) on the same navigation problem. A cell is black if it is blocked and this fact is known to the agent. A ReusableTree-pointer is shown as a thick arrow if it was added to the reusable path (or reusable tree) in the current A* search and as a thin arrow if it was added in a previous A* search. A triangle marks the cell that the current A* search was about to expand before it terminated.

Figures 4(a) and 4(e) show that the first A* searches of Path-AA* and Tree-AA* produce the same result. The agent then moves from E2 to E3, where it senses that E4 is blocked. Path-AA* removes E2 E3 E4 E5 E6 E7 E8 E9 D9 from the reusable path. Figure 4(b) shows that the second A* search of Path-AA* terminates when it is about to expand D9, and Path-AA* adds E3 D3 D4 D5 D6 D7 D8 D9 (shown as thick arrows) to the reusable path. On the other hand, Tree-AA* removes E2 E3 E4 E5 from the reusable tree. Figure 4(f) shows that the second A* search of Tree-AA* also terminates when it is about to expand D9, but Tree-AA* adds E3 D3 D4 D5 D6 D7 D8 D9 (shown as thick arrows) to the reusable tree. Thus, Tree-AA* removes fewer cells, which might allow its future A* searches to terminate earlier. The agent then moves from E3 to D3, where it senses that D4 is blocked. Path-AA* and Tree-AA* perform their third A* searches. The agent then moves from

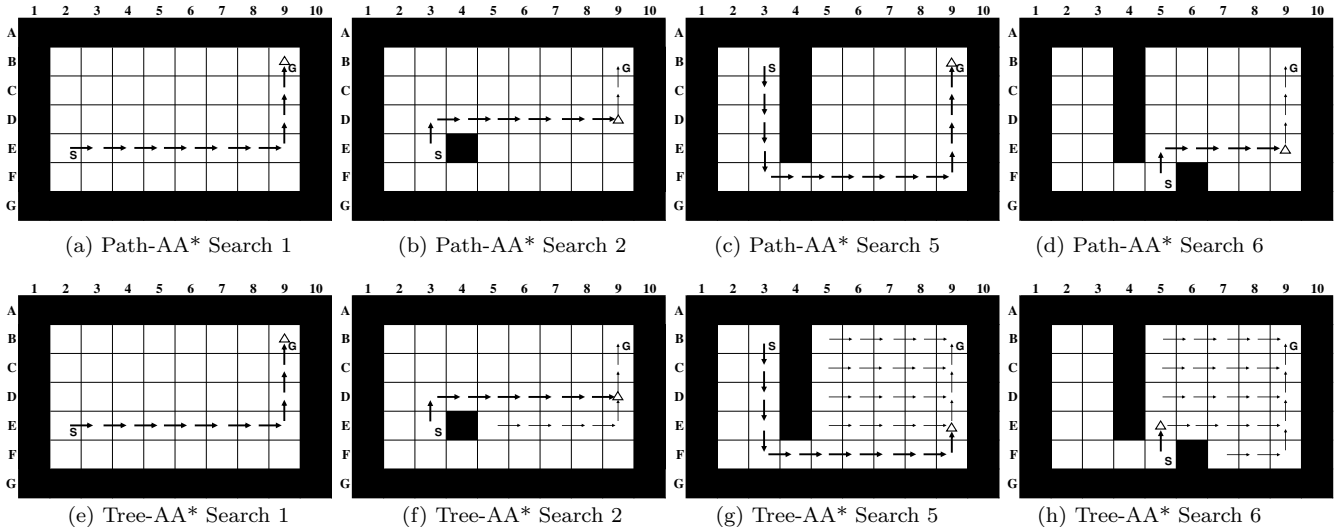


Figure 4: Comparison of Path-AA* (top) and Tree-AA* (bottom)

D3 to C3, where it senses that C4 is blocked. Path-AA* and Tree-AA* perform their fourth A* searches. We omit the details of the third and fourth A* searches due to space constraints. The agent then moves from C3 to B3, where it senses that B4 is blocked. Figure 4(c) shows that the fifth A* search of Path-AA* terminates when it is about to expand B9. Figure 4(g) shows that the fifth A* search of Tree-AA* terminates already when it is about to expand E9, illustrating the speed up resulting from reusing the paths from all previous A* searches. The agent then moves from B3 to F5, where it senses that F6 is blocked. Figure 4(d) shows that the sixth A* search of Path-AA* terminates when it is about to expand E9. Figure 4(h) shows that the fifth A* search of Tree-AA* terminates already when it is about to expand F5, again illustrating the speed up resulting from reusing the paths from all previous A* searches.

5. TREE-AA*-BACK

An A* search with consistent h-values guarantees that the g-value of every expanded state is equal to the cost of a minimum-cost path from the start state to the expanded state [17]. Thus, if the first A* search of Tree-AA* searches from the goal state to the current state of the agent (= backward search), then the resulting search tree restricted to the expanded states is a reusable tree. All subsequent A* searches of Tree-AA* must be forward searches. We refer to the resulting version of Tree-AA* as Tree-AA*-Back. The reusable tree after the first A* search of (standard) Tree-AA* contains only the expanded states on the minimum-cost path from the current state of the agent to the goal state, while the reusable tree after the first A* search of Tree-AA*-Back contains all expanded states, which might allow future A* searches to terminate earlier.

6. EXPERIMENTAL EVALUATION

We compare Tree-AA* to the top incremental heuristic search algorithms in the context of goal-directed navigation in unknown grids, namely Adaptive A* (that uses forward searches, which is the most efficient search direction for Adaptive A*), Path-AA* (that uses forward searches, which is the only possible search direction for Path-AA*)

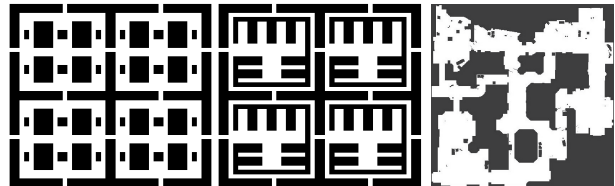


Figure 5: Maps

and D* Lite (that uses backward searches, which is the only possible search direction for D* Lite). Tree-AA* uses forward searches but we also implement Tree-AA*-Back from Section 5, whose first A* search is a backward search. For fairness, all search algorithms use binary heaps as priority queues and break ties among states with the same f-values in favor of states with larger g-values (which is known to be a good tie-breaking strategy), with one exception: During the first search, Tree-AA*-Back (1) breaks ties among states with the same f-values in favor of states with larger g-values but Tree-AA*-Back (2) breaks ties in favor of states with smaller g-values. During the remaining A* searches, both versions of Tree-AA*-Back break ties in favor of states with larger g-values.

6.1 Experimental Setup

We used four-neighbor grids as examples since they result in integer-valued g-values, h-values and f-values. We use eight-neighbor grids in the experiments since they are often preferred in practice, for example in video games [3, 2]. The agent always senses the blockage status of its eight neighboring cells and can then move to any one of the unblocked neighboring cells with cost one for horizontal or vertical movements and cost $\sqrt{2}$ for diagonal movements. The user-given h-values are the octile distances [3].

We use two indoor office maps of size $1,000 \times 1,000$ cells, where the size of each room is 20×20 cells. Figure 5 (left and center) shows areas of 2×2 rooms in office maps 1 and 2. We also use a computer game map of size $3,000 \times 3,000$ cells adapted from Counter-Strike (courtesy of Vadim Bulitko from the University of Alberta). Figure 5 (right) shows the game map. We average our experimental results

Office Map 1									
	All A* Searches				First A* Search		Remaining A* Searches		
	(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
Adaptive A*	1,333	652	473	0.127	82.8	900	0.48	472	0.126
D* Lite	1,520	474	562	0.177	83.9	25,836	9.25	508	0.158
Path-AA*	1,333	652	114	0.043	28.0	900	0.48	113	0.042
Tree-AA*	1,333	652	13	0.005	3.3	900	0.45	11	0.004
Tree-AA*-Back (1)	1,337	615	55	0.015	9.2	25,836	5.84	13	0.006
Tree-AA*-Back (2)	1,334	652	212	0.059	38.5	134,669	36.42	6	0.003
Office Map 2									
Adaptive A*	6,606	3,924	512	0.130	445.8	900	0.47	512	0.130
D* Lite	5,821	2,429	285	0.092	223.5	48,013	17.14	265	0.085
Path-AA*	6,637	3,094	88	0.033	102.1	900	0.47	88	0.033
Tree-AA*	6,637	3,094	19	0.006	18.6	900	0.48	19	0.006
Tree-AA*-Back (1)	6,413	2,903	36	0.008	23.2	48,013	10.77	19	0.004
Tree-AA*-Back (2)	6,635	3,088	60	0.016	49.4	132,795	36.11	17	0.004
Game Map									
Adaptive A*	4,842	2,524	3,424	1.094	2,761.3	2,671	1.62	3,424	1.094
D* Lite	5,723	2,491	1,547	0.790	1,967.9	5,821	3.64	1,545	0.789
Path-AA*	4,841	2,520	1,442	0.525	1,323.0	2,671	1.65	1,442	0.525
Tree-AA*	4,841	2,519	1,353	0.437	1,100.8	2,671	1.57	1,353	0.437
Tree-AA*-Back (1)	4,955	2,378	1,320	0.407	967.8	5,821	2.29	1,322	0.406
Tree-AA*-Back (2)	4,841	2,519	1,610	0.466	1,173.9	1,203,590	392.12	1,132	0.310

(a) = moves per test case; (b) = A* searches per test case; (c) = cell expansions per A* search; (d) = runtime per A* search; (e) = runtime per test case; (f) = cells expansions of the 1st A* search; (g) = runtime of the 1st A* search; (h) = cell expansions per A* search (excluding the 1st A* search); (i) = runtime per A* search (excluding the 1st A* search).

Table 1: Experimental Results

over 500 test cases with a reachable goal cell for each map. For each test case in the office maps, we ensure that the start and goal cells are far apart by independently choosing the x-coordinate of the start cell randomly between 1 and 100 and the x-coordinate of the goal cell randomly between 901 and 1,000. We independently choose the y-coordinates of the start and goal cells randomly between 1 and 1,000. Similarly, for each test case in the game map, we independently choose the x-coordinate of the start cell randomly between 1 and 300 and the x-coordinate of the goal cell randomly between 2,701 and 3,000. We independently choose the y-coordinates of the start and goal cells randomly between 1 and 3,000.

6.2 Experimental Results

We report two measures of the difficulty of goal-directed navigation problems in unknown grids, namely (a) the number of moves of the agent per test case and (b) the number of A* searches per test case until the agent reaches the goal cell. These measures vary slightly among the compared search algorithms since they can determine different minimum-cost paths, in which case the agents that follow the paths might sense different blocked cells, which can make their trajectories diverge. We report three measures of the efficiency of the search algorithms, namely (c) the number of expanded cells per A* search, (d) the runtime per A* search in milliseconds and (e) the runtime per test case in milliseconds on a Linux PC with a Pentium CoreQuad 2.33 GHz CPU and 8 GB RAM. Since the number of A* searches is (approximately) the same for all search algorithms, their runtimes per test case are largely proportional to their runtimes per A* search. Therefore, the main measure of the efficiency of the search algorithms is their runtime per A* search. In order to gain more insight into the behavior of the search algorithms, we divide each test case into two parts, namely the first A* search and the remaining A* searches other than the first one. For the first A* search, we report (f) the number of expanded cells and (g) the runtime in milliseconds. For the remaining A* searches other than the first one, we report (h) the number of expanded cells per A* search and (i) the runtime per A* search in milliseconds. Table 1 shows the following relationships:

First, Tree-AA* has a smaller runtime per A* search than Adaptive A* for all maps because it has a smaller number of cell expansions per A* search due to the speed up achieved with the termination principle. For example, Tree-AA* expands only about 2.7, 3.7 and 39.5 percent of the cells per

A* search that Adaptive A* expands in office maps 1 and 2 and the game map, respectively. It thus runs by factors of 25.4, 21.7 and 2.5 faster per A* search.

Second, Tree-AA* has a smaller runtime per A* search than D* Lite for all maps, which is due to two reasons. First, Tree-AA* has a smaller number of cell expansions per A* search perhaps due to the speed up achieved with the update and termination principles. For example, Tree-AA* expands only about 2.3, 6.7 and 87.5 percent of the cells per A* search that D* Lite expands in office maps 1 and 2 and the game map, respectively. Second, Tree-AA* has a smaller number of heap percolates per A* search due to both the smaller number of cell expansions per A* search (resulting in fewer heap operations) and a smaller number of cells in the open list during each A* search (resulting in fewer heap percolates per heap operation). The smaller number of cells in the open list is due to each A* search of Tree-AA* starting with an empty open list rather than the open list at the end of the previous A* search. Tree-AA* thus runs by factors of 35.4, 15.3 and 1.8 faster per A* search.²

Third, Tree-AA* has a smaller runtime per A* search than Path-AA* for all maps because it has a smaller number of cell expansions per A* search due to the speed up achieved with a reusable tree rather than a reusable path. For example, Tree-AA* expands only about 11.4, 21.6 and 93.8 percent of the cells per A* search that Path-AA* expands in office maps 1 and 2 and the game map, respectively. It thus runs by factors of 8.6, 5.5 and 1.2 faster per A* search.

Fourth, both versions of Tree-AA*-Back have a larger runtime for the first A* search than (standard) Tree-AA* for all maps but tend to have a smaller runtime per A* search for the remaining A* searches, which is due to the following reasons: The first A* search of Tree-AA* is a forward search, while the first A* search of both versions of Tree-AA*-Back

²To understand better in which situations Tree-AA* has an advantage over D* Lite, we also perform experiments on maps of size 500×500 cells and independently block 20, 30, 40, 50 and 60 percent of randomly chosen cells, respectively. We average our experimental results over 500 test cases with a reachable goal cell for each map. For each test case, we choose the x-coordinate of the start cell randomly between 1 and 50 and the x-coordinate of the goal cell randomly between 451 and 500. We independently choose the y-coordinates of the start and goal cells randomly between 1 and 500. Tree-AA* expands only about 11.2, 12.5, 14.9, 17.4 and 87.0 percent of the cells per A* search that D* Lite expands, respectively.

is a backward search. Backward A* searches expand more cells than forward A* searches [10]. Thus, the runtime of the first A* search of both versions of Tree-AA*-Back is larger than the one of Tree-AA*. The first A* search of Tree-AA* yields a reusable path, while the first A* search of both versions of Tree-AA*-Back yields a reusable tree. During the first search, Tree-AA*-Back (1) breaks ties among states with the same f-values in favor of states with larger g-values but Tree-AA*-Back (2) breaks ties in favor of states with smaller g-values. Thus, the runtime of the first A* search and the size of the reusable tree of Tree-AA*-Back (2) are larger than the ones of Tree-AA*-Back (1). The larger the reusable tree, the more it speeds up the first few remaining A* searches of Tree-AA*-Back due to the termination principle until the reusable trees of Tree-AA*-Back and Tree-AA* are about equally large. For example, Tree-AA*-Back (1) and, given in parentheses, Tree-AA*-Back (2) expand about 423.1 (1630.8), 189.5 (315.8) and 97.6 (119.0) percent of the cells per A* search that Tree-AA* expands in office maps 1 and 2 and the game map, respectively. They thus run by factors of 3.0 (11.8), 1.3 (2.7), and 0.9 (1.1) more slowly per A* search. However, Tree-AA*-Back (1) and (2) expand about 118.2 (54.5), 100.0 (89.5) and 97.7 (83.7) percent of the cells per A* search that Tree-AA* expands for the remaining A* searches other than the first one. They thus run by factors of 0.7 (1.3), 1.5 (1.5) and 1.1 (1.4) faster per A* search for the remaining A* searches. Therefore, Tree-AA*-Back (1) and (2) tend to run faster than Tree-AA* for applications where the first A* search can be performed offline before the goal-directed navigation problem in unknown terrain starts and its runtime thus does not matter, as is often the case in robotics.

7. CONCLUSIONS

In this paper, we introduced a new incremental heuristic search algorithm called Tree Adaptive A*. So far, there existed incremental heuristic search algorithms (such as Adaptive A*) that make the h-values of the current A* search more informed and incremental heuristic search algorithms (such as D* Lite) that change the search tree of the current A* search to the search tree of the next A* search. Tree Adaptive A* uses the update principle of Adaptive A* to make the h-values of the current A* search more informed. It also uses the termination principle of Path Adaptive A* to terminate A* searches earlier than regular A* searches but generalizes it to reuse suffixes of the minimum-cost paths of the current and all previous A* searches (= reusable tree). Overall, Tree Adaptive A* is the first incremental heuristic search algorithm to combine the principles of both classes of incremental heuristic search algorithms and can run faster than Adaptive A*, Path Adaptive A* and D* Lite, the top incremental heuristic search algorithms in the context of goal-directed navigation in unknown grids.

8. REFERENCES

- [1] M. Björnsson, M. Enzenberger, R. Holte, J. Schaeffer, and P. Yap. Comparison of different abstractions for pathfinding on maps. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1511–1512, 2003.
- [2] V. Bulitko, Y. Björnsson, M. Luvstrek, J. Schaeffer, and S. Sigmundarson. Dynamic control in path-planning with real-time heuristic search. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 49–56, 2007.
- [3] V. Bulitko and G. Lee. Learning in real-time search: A unifying framework. *Journal of Artificial Intelligence Research*, 25:119–157, 2006.
- [4] H. Choset, S. Thrun, L. Kavraki, W. Burgard, and K. Lynch. *Principles of Robot Motion: Theory, Algorithms, and Implementations*. MIT Press, 2005.
- [5] D. Ferguson and A. Stentz. Using interpolation to improve path planning: The Field D* algorithm. *Journal of Field Robotics*, 23(2):79–101, 2006.
- [6] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 2:100–107, 1968.
- [7] C. Hernández, P. Meseguer, X. Sun, and S. Koenig. Path-Adaptive A* for incremental heuristic search in unknown terrain [short paper]. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 358–361, 2009.
- [8] R. Holte, M. Perez, R. Zimmer, and A. MacDonald. Hierarchical A*: Searching abstraction hierarchies efficiently. In *Proceedings of the National Conference on Artificial Intelligence*, pages 530–535, 1996.
- [9] S. Koenig, D. Furcy, and C. Bauer. Heuristic search-based replanning. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 294–301, 2002.
- [10] S. Koenig and M. Likhachev. Fast replanning for navigation in unknown terrain. *Transaction on Robotics*, 21(3):354–363, 2005.
- [11] S. Koenig and M. Likhachev. A new principle for incremental heuristic search: Theoretical results. In *Proceedings of the International Conference on Autonomous Planning and Scheduling*, pages 410–413, 2006.
- [12] S. Koenig, M. Likhachev, Y. Liu, and D. Furcy. Incremental heuristic search in artificial intelligence. *Artificial Intelligence Magazine*, 25(2):99–112, 2004.
- [13] S. Koenig, C. Tovey, and Y. Smirnov. Performance bounds for planning in unknown terrain. *Artificial Intelligence*, 147(1-2):253–279, 2003.
- [14] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun. Anytime Dynamic A*: An anytime, replanning algorithm. In *Proceedings of the International Conference on Automated Planning and Scheduling*, pages 262–271, 2005.
- [15] M. Likhachev, D. Ferguson, G. Gordon, A. Stentz, and S. Thrun. Anytime search in dynamic graphs. *Artificial Intelligence*, 172(14):1613–1643, 2008.
- [16] K. Matsuta, H. Kobayashi, and A. Shinohara. Multi-target Adaptive A*. In *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, pages 1065–1072, 2010.
- [17] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1985.
- [18] A. Stentz. The Focussed D* algorithm for real-time replanning. In *Proceedings of International Joint Conference in Artificial Intelligence*, pages 1652–1659, 1995.
- [19] X. Sun, S. Koenig, and W. Yeoh. Generalized Adaptive A*. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 469–476, 2008.

Distributed Problem Solving I

Quality guarantees for region optimal DCOP algorithms

Meritxell Vinyals¹, Eric Shieh², Jesus Cerquides¹, Juan Antonio Rodriguez-Aguilar¹,
Zhengyu Yin², Milind Tambe², and Emma Bowring³

¹ Artificial Intelligence Research Institute (IIIA), Campus UAB, Bellaterra, Spain

{meritxell, cerquide,jar}@iiia.csic.es

² University of Southern California, Los Angeles, CA 90089

{eshieh, zhengyuy,tambe}@usc.edu

³ University of the Pacific, Stockton, CA 95211

ebowring@pacific.edu

ABSTRACT

k - and t -optimality algorithms [9, 6] provide solutions to DCOPs that are optimal in regions characterized by its size and distance respectively. Moreover, they provide quality guarantees on their solutions. Here we generalise the k - and t -optimal framework to introduce \mathcal{C} -optimality, a flexible framework that provides reward-independent quality guarantees for optima in regions characterised by any arbitrary criterion. Therefore, \mathcal{C} -optimality allows us to explore the space of criteria (beyond size and distance) looking for those that lead to better solution qualities. We benefit from this larger space of criteria to propose a new criterion, the so-called size-bounded-distance criterion, which outperforms k - and t -optimality.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed AI

General Terms

Algorithms, Design, Theory

Keywords

DCOP, approximate algorithm, bound, region optimality

1. INTRODUCTION

Distributed Constraint Optimization (DCOP) is a popular framework for cooperative multi-agent decision making. It has been applied to real-world domains such as sensor networks [14], traffic control [5], or meeting scheduling [10]. In real-world domains, and particularly in large-scale applications, DCOP techniques have to cope with limitations on resources and time available for reasoning. Because DCOP is NP-Hard [8], complete DCOP algorithms (e.g. Adopt [8], OptAPO [7], DPOP [10]) that guarantee global optimality are unaffordable for these domains due to their exponen-

tial costs. In contrast to complete algorithms, incomplete algorithms [14, 4, 12, 9, 6] provide better scalability.

Unfortunately, an important limitation for the application of incomplete algorithms is that they usually fail to provide quality guarantees on their solutions. The importance of quality guarantees is twofold. First, they help guarantee that agents do not converge to a solution whose quality is below a certain fraction of the optimal solution (which can have catastrophic effects in certain domains). Secondly, quality guarantees can aid in algorithm selection and network structure selection in situations where the algorithmic cost of coordination must be weighed up against solution quality (trade-off cost versus quality).

To the best of our knowledge, k -size and t -distance optimal algorithms [9, 6] are the only incomplete DCOP algorithms that can provide guarantees on the worst-case solution quality of their solutions at design time and exploiting different levels of knowledge of the particular problem instance(s). These quality guarantees exploit the available knowledge, if any, about the DCOP(s) to solve regarding their graph structure [9] and their reward structure [3]. Unlike other incomplete algorithms that focus on individual agent decisions [14, 4, 12], k -size [9] and t -distance [6] optimal algorithms are based on coordinating the decisions of local groups (neighbourhoods) of agents. Thus, given a DCOP, agents inside a neighbourhood coordinate to locally optimise their joint decision by considering any joint assignment that can improve their joint reward. The difference between k -size and t -distance optimal algorithms is the criterion employed to generate neighbourhoods: k -size-optimality creates neighbourhoods of a fixed size (k), whereas t -distance-optimality creates per each agent a neighbourhood that includes all other agents within a certain distance (t) in the constraint graph. In both cases, we can regard a collection of neighbourhoods as an *exploration region* for either a k - or t -optimal algorithm in a constraint graph.

Although k -size and t -distance are the criteria explored so far in the literature, it is reasonable to wonder whether there are further local optimality criteria that can lead to better solution qualities while providing quality guarantees. In this paper we provide the foundations to explore this fundamental research question. First of all, we generalise the k - and t -optimal framework to introduce \mathcal{C} -optimality, a flexible framework that provides quality guarantees for local

Cite as: Quality guarantees for region optimal DCOP algorithms, M. Vinyals et al., *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 133-140. Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

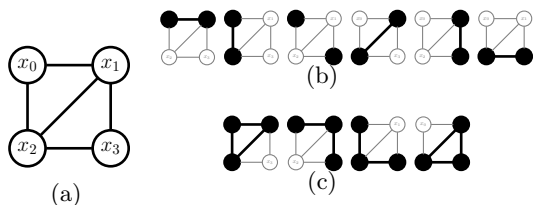


Figure 1: Example of (a) a DCOP graph, (b) its 2-size region and (c) its 3-size region.

optima in regions characterised by any arbitrary criterion. Therefore, \mathcal{C} -optimality allows us to explore the space of local optimality criteria (beyond size and distance) looking for those that lead to better solution qualities.

We benefit from this larger space of criteria to propose a novel criterion to define regions, the so-called size-bounded-distance criterion, which we design to overcome the main drawbacks of size and distance optimality. Finally, we extend the DALO algorithm proposed in [6] to compute \mathcal{C} -optimal solutions. We empirically show that the size-bounded-distance criterion indeed leads to better solution qualities, outperforming k - and t -optimality. Therefore, the \mathcal{C} -optimality framework opens new research opportunities to study the design of new local optimality criteria.

Likewise k - and t - optimality, \mathcal{C} -optimality is algorithmic-independent, meaning that bounds are defined over solutions not over the algorithms to find them. Thus, although we propose \mathcal{C} -DALO as general purpose algorithm to find any \mathcal{C} -optimal for arbitrary criteria, it does not imply that there cannot exist different region-optimal algorithms. Indeed, in our previous work [11], we employ analogous guarantees to provide worst-case bounds on the solutions of the loopy belief propagation algorithm, a popular approximate algorithm for finding the Maximum a Posteriori assignment in Markov Random Fields. In contrast, this work focuses on DCOPs. Firstly, we generalise the bounds in [11] to provide a framework for regional DCOPs algorithms. Secondly, we analyse the benefits of exploring arbitrary region criteria.

The paper is organised as follows. Section 2 provides some background on DCOPs and on the k - and t -optimality frameworks. Section 3 introduces the notion of \mathcal{C} -optimality solution as a local solution for an arbitrary criterion and the mechanisms for computing quality guarantees for \mathcal{C} -optimal solutions. Moreover, it also proves that the \mathcal{C} -optimality framework generalises k - and t -optimality. Section 4 introduces a new local optimality criterion, size-bounded distance, and empirically compares the quality solutions obtained by the new criterion with respect to k - and t -optimal solutions. Finally, section 5 draws conclusions and sets paths to future research.

2. BACKGROUND

2.1 DCOP Definition

A Distributed Constraint Optimization Problem (DCOP) consists of a set of variables, each assigned to an agent which must assign a discrete value to the variable: these values correspond to individual actions that can be taken by agents. Constraints exist between subsets of these variables that determine rewards to the agent team based on the combinations of values chosen by their respective agents, namely relations. Let $\mathcal{X} = \{x_1, \dots, x_n\}$ be a set of variables over

domains $\mathcal{D}_1, \dots, \mathcal{D}_n$. A relation on a set of variables $V \subseteq \mathcal{X}$ is expressed as a reward function $S_V : \mathcal{D}_V \rightarrow \mathbb{R}^+$, where \mathcal{D}_V is the joint domain over the variables in V . This function represents the reward generated by the relation over the variables in V when the variables take on an assignment in the joint domain \mathcal{D}_V . Whenever there is no need to identify the domain, we simply use S to note relations.

In a DCOP each agent knows all the relations that involve its variable(s). In this work we assume that each agent is assigned a single variable, so we will use the terms “agent” and “variable” interchangeably.

Formally, a DCOP is a tuple $\langle \mathcal{X}, \mathcal{D}, \mathcal{R} \rangle$, where: \mathcal{X} is a set of variables (each one assigned to a different agent); \mathcal{D} is the joint domain space for all variables; and \mathcal{R} is a set of reward relations. The solution quality for an assignment $d \in \mathcal{D}$ to the variables in \mathcal{X} is the sum of the rewards for the assignment over all the relations in the DCOP, namely:

$$R(d) = \sum_{S_V \in \mathcal{R}} S_V(d_V) \quad (1)$$

where $d_V \in \mathcal{D}_V$ contains the values assigned by d to the variables in V . With slight abuse of notation we allow to write equation 1 as $R(d) = \sum_{S \in \mathcal{R}} S(d)$.

Solving a DCOP amounts to choosing values for the variables in \mathcal{X} such that the solution quality is maximized. A binary DCOP (each relation involves a maximum of two variables) is typically represented by its constraint graph, whose vertices stand for variables and whose edges link variables that have some direct dependency (appear together in the domain of some relation). Examples of constraint graphs are depicted in figures 1(a) and 2(a).

2.2 Size and distance optimality

Since DCOP is NP-hard, an important line of work focuses on developing fast incomplete algorithms. Along this direction, a significant trend is to study approaches based on coordinating the decisions of local groups of agents, instead of having each agent make an individual choice.

Two important local optimality criteria that establish how to group agents to coordinate their decisions are k -size [9] and t -distance [6] optimality. According to k -size optimality agents form groups of k agents. For instance, figures 1(b) and 1(c) depict the groups of 2 agents and 3 agents respectively for the DCOP in figure 1(a) where boldfaced nodes stand for agents included in the group. Given an assignment x^* , it is a local optimum, k -optimum, when no group of k or fewer agents can improve its reward $R(x^*)$ by simultaneously changing their variable assignments. On the other hand, t -optimality defines locality based on a group of surrounding nodes within a fixed distance t of a central node. For instance, figures 2(b) and 2(c) depict the groups of agents at distance 1 and 2 respectively for each agent in the DCOP in figure 2(a). Likewise k -optimality, a t -optimum occurs when no group of agents can improve its reward.

k - and t -optimal algorithms represent an important class of incomplete algorithms that have agents dynamically form local groups to coordinate action choices. A significant feature of k - and t -optimal algorithms is that they provide guarantees on the solution quality of a DCOP as a fraction of the global optimum, prior to the execution of the algorithm. An algorithm has a quality guarantee δ (being $0 \leq \delta \leq 1$) if every solution provided by the algorithm has at least quality $\delta \cdot R(x^*)$ where x^* stands for the global optimum assignment.

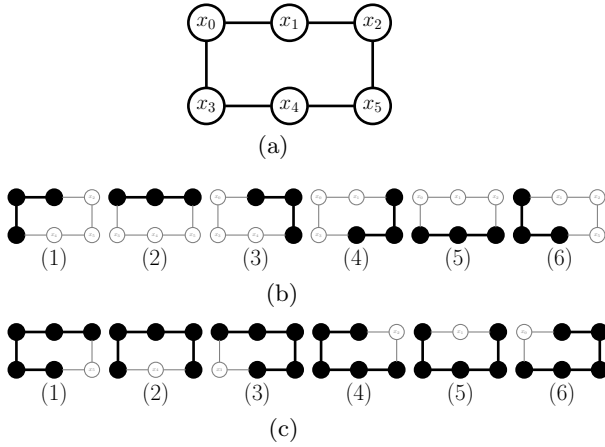


Figure 2: Example of (a) a DCOP graph, (b) its 1-distance region and (c) its 2-distance region.

The larger the quality guarantee, the closer the algorithm is to providing an optimal solution to the problem.

Both k - and t -optimality have explored mechanisms for computing bounds. Firstly, both k -size optimality and t -distance optimality provide means for computing bounds independently of the problem instance [9, 6], namely disregarding the graph structure and reward structure. Secondly, knowledge of a problem instance can be used to obtain tighter guarantees. One way is to exploit the knowledge about the graph structure of the DCOP (e.g. star, ring) [9]. Another way is to exploit the reward structure [3]. We can group such mechanisms based on their computational costs.

On the one hand, a tight bound on the quality of every k - or t -optimum can be computed using a linear program (LP) [9, 6]. In this method, rewards on the relations in the DCOP are treated as variables in a program whose goal is to minimise the quality guarantee. When the program is solved, the decision variables are instantiated with the values that, if used as relation rewards, would produce the DCOP whose local optimum has the lowest reward with respect to the global optimal solution. For example, for k -optimality and for a specific graph structure, after running the program we obtain (1) a quality guarantee δ for any k -optimal solution on any DCOP having the specific constraint graph and (2) a DCOP having the specific constraint graph and a k -optimal solution x^k whose quality is equal to the bound, namely $R(x^k) = \delta \cdot R(x^*)$.

On the other hand, there are methods that are computationally cheaper and can compute bounds in constant time [9, 6]. Despite the computational savings of these methods, with respect to the LP-based approach, in general tightness is not guaranteed.

3. GENERALIZING SIZE AND DISTANCE OPTIMALITY

In this section we generalize the concept of size and distance optimality to \mathcal{C} -optimality, which allows us to characterize any local optimum in a region \mathcal{C} characterized by an arbitrary criterion.

Notice that given a DCOP both k - and t - local optimality criteria define a region, namely a family of neighbourhoods (subsets of variables) \mathcal{C} . For instance, in figure 2(b), we show the neighbourhoods in the 1-distance region of the

DCOP in figure 2(a), where boldfaced nodes in the constraint graph stand for variables included in the neighbourhood. Given some assignment x , we say that it is optimal in a neighbourhood $\mathcal{C}^\alpha \in \mathcal{C}$ if its reward cannot be improved by changing the values of some of the variables in the neighbourhood. For instance, the first graph on the left in figure 2(b) represents a neighbourhood. An assignment x is optimal in that neighbourhood if any other assignment that maintains the values of x_2, x_4 and x_5 receives at most the same reward as x . Then, we can claim optimality for x in a region \mathcal{C} (noted as $x^{\mathcal{C}}$) whenever it is optimal in each neighbourhood in the region. For instance, an assignment x will be optimal in the region depicted in figure 2(c) if it is optimal in each of its neighbourhoods. Therefore, in general, for both k -size and t -distance based optimality, we observe that:

- each criterion is based on the definition of a region over the constraint graph; and
- given any assignment, checking for either k -size or t -distance optimality amounts to checking for optimality in that region.

Hereafter we propose a general notion of region optimality, the so-called \mathcal{C} -optimality, and describe how to calculate bounds for a \mathcal{C} -optimal assignment, namely an assignment that is optimal in an arbitrary region \mathcal{C} .

3.1 Region optimality

Next, we introduce the concepts of neighbourhood and region so that we can formally define \mathcal{C} -optimality. After that, we analyse the way in which neighbourhoods relate to each other by formalizing the idea that a larger neighbourhood covers a smaller one.

Formally, a neighbourhood is a subset of variables of \mathcal{X} . Given two assignments x and y , we define $D(x, y)$ as the set containing the variables whose values in x and y differ. Given a neighbourhood A , we say that x is a *neighbour of y in A* iff x differs from y only in variables that are contained in A .

A region \mathcal{C} is a multi-set¹ of subsets of \mathcal{X} , namely a multi-set of neighbourhoods of \mathcal{X} . Given a region \mathcal{C} , we say that x is *inside region \mathcal{C} of y* iff x differs from y only in variables that are contained in one of the neighbourhoods in \mathcal{C} , that is, if there is a neighborhood $\mathcal{C}^\alpha \in \mathcal{C}$ such that x is neighbour of y in \mathcal{C}^α .

An assignment x is \mathcal{C} -optimal if it cannot be improved by any other assignment inside region \mathcal{C} of x . That is, for every assignment y inside region \mathcal{C} of x , we have that $R(x) \geq R(y)$.

Relations among neighbourhoods

Given two neighbourhoods $A, B \subseteq \mathcal{X}$ we say that B completely covers A if $A \subseteq B$. We say that B does not cover A at all if $A \cap B = \emptyset$. Otherwise, we say that B covers A partially.

As an example of these relations, consider neighbourhoods (1) and (4) in figure 2(b), noted as $A = \{x_0, x_1, x_3\}$ and $B = \{x_2, x_4, x_5\}$ respectively, and neighbourhood (1) in figure 2(c), noted as $C = \{x_0, x_1, x_2, x_3, x_4\}$. Then, we have that A covers C partially (it contains some variables in C) whereas C covers A completely (C contains all variables in

¹A multi-set is a generalisation of a set that can hold multiple instances of the very same element.

A). Moreover, A does not cover B at all and vice versa because these neighbourhoods do not have any variable in common.

Then, we say that $A \subseteq \mathcal{X}$ is covered by \mathcal{C} if there is a neighbourhood $C^\alpha \in \mathcal{C}$ such that C^α completely covers A . For example, neighbourhood (1) in figure 2(b) is covered by the region of neighbourhoods in figure 2(c), because, among others, neighbourhood (1) in this region covers it completely.

For each neighbourhood C^α we can classify each relation S in a DCOP into one of three disjoint groups, depending on whether C^α covers S completely ($T(C^\alpha)$), partially ($P(C^\alpha)$), or not at all ($N(C^\alpha)$).

For each relation $S_V \in \mathcal{R}$ we define $cc(S_V, \mathcal{C}) = |\{C^\alpha \in \mathcal{C} \text{ s.t. } V \subseteq C^\alpha\}|$, that is, the number of neighbourhoods in \mathcal{C} that cover the domain of S_V completely. We also define $nc(S_V, \mathcal{C}) = |\{C^\alpha \in \mathcal{C} \text{ s.t. } V \cap C^\alpha = \emptyset\}|$, that is, the number of neighbourhoods in \mathcal{C} that do not cover the domain of S_V at all.

3.2 Quality guarantees for region optima

After its formal definition, we are interested in providing a bound on the quality of any \mathcal{C} -optimal assignment in a DCOP with non-negative rewards. We say that we have a bound δ when we can state that the quality of any \mathcal{C} -optimal assignment x^C is larger than δ times the quality of the optimal x^* . Hence, having a bound δ means that for every x^C we have that $\frac{R(x^C)}{R(x^*)} \geq \delta$. For a given set of relations \mathcal{R} , let x^C be the \mathcal{C} -optimal assignment with smallest reward, then $\frac{R(x^C)}{R(x^*)}$ provides a tight bound on the quality of any \mathcal{C} -optimal assignment for the specific rewards \mathcal{R} .

We are interested in defining bounds that are independent of the particular reward values of the DCOP. In that setting, a simple way to provide a bound on the quality is to directly search the space of reward values to find the set of rewards \mathcal{R}^* that minimizes $\frac{R^*(x^C)}{R^*(x^*)}$.

More formally, this can be encoded as:

Find \mathcal{R} , x^C and x^* that

$$\text{minimize } \frac{R(x^C)}{R(x^*)}$$

subject to x^C being a \mathcal{C} -optimal for \mathcal{R}

Applying some transformations detailed in [13], we can simplify this program into the following linear program (LP) with x and y being vectors of positive real numbers:

$$\text{minimize } \sum_{S \in \mathcal{R}} x_S$$

subject to

$$\sum_{S \in \mathcal{R}} y_S = 1$$

and for each neighbourhood C covered by \mathcal{C} subject to

$$\sum_{S \in \mathcal{R}} x_S \geq \sum_{S \in T(C)} y_S + \sum_{S \in N(C)} x_S$$

where $T(C)$ contains the relations completely covered by C and $N(C)$ the relations that are not covered by C at all.

After solving this LP, $\delta = \sum_{S \in \mathcal{R}} x_S$ provides a tight bound on the quality of a \mathcal{C} -optimal solution for the graph structure represented by \mathcal{R} . Let M be the number of variables of the largest neighbourhood in \mathcal{C} . The LP has $2 \cdot |\mathcal{R}|$ variables and $\mathcal{O}(2^M \cdot |\mathcal{C}|)$ constraints, and hence it is solvable in time polynomial in $|\mathcal{R}|$ and in $2^M \cdot |\mathcal{C}|$.

3.3 Faster quality guarantees

The computational complexity of the previous LP can be high as the number of relations $|\mathcal{R}|$, the number of neighbourhoods $|\mathcal{C}|$ or its size M grows. In this section we show that we can compute a bound in time $\mathcal{O}(|\mathcal{R}||\mathcal{C}|)$. Further-

more, the result will prove as a very valuable tool for future theoretical developments. As a counterpart, we lose the tightness of the bound.

PROPOSITION 1. *Let $(\mathcal{X}, \mathcal{D}, \mathcal{R})$ be a DCOP with non-negative rewards and \mathcal{C} a region. If x^C is a \mathcal{C} -optimal assignment then*

$$R(x^C) \geq \frac{cc_*}{|\mathcal{C}| - nc_*} R(x^*) \quad (2)$$

where $cc_* = \min_{S \in \mathcal{R}} cc(S, \mathcal{C})$, $nc_* = \min_{S \in \mathcal{R}} nc(S, \mathcal{C})$, and x^* is the optimal assignment.

Proposition 1 directly provides a simple algorithm to compute a bound. Given a region \mathcal{C} and a graph structure, we can directly assess cc_* and nc_* by computing $cc(S, \mathcal{C})$ and $nc(S, \mathcal{C})$ for each relation $S \in \mathcal{R}$ and taking the minimum. This will take time $\mathcal{O}(|\mathcal{R}||\mathcal{C}|)$, that is linear in the number of relations of the DCOP and linear in the number of neighbourhoods in the region.

As an example, now we turn back to figure 2 to assess the bounds for a \mathcal{C} -optimal assignment using equation 2. First, we assess the bound for the 1-distance region \mathcal{C}_1 in figure 2(b). Given the relation $S = R_{\{x_0, x_1\}}$, we assess the number of neighbourhoods that completely cover $\{x_0, x_1\}$ as $cc(S, \mathcal{C}_1) = 2$ (the two first neighbourhoods on the left-hand side) and the number of neighbourhoods that do not cover $\{x_0, x_1\}$ at all as $nc(S, \mathcal{C}_1) = 2$ (the fourth and fifth neighbourhoods). After repeating the process for the rest of relations in the constraint graph, we obtain that $cc_* = 2$ and $nc_* = 2$, and hence $\frac{cc_*}{|\mathcal{C}_1| - nc_*} = \frac{2}{6-2} = \frac{1}{2}$. Notice that this leads to a better bound than the one we obtain following the result in [6], since $\frac{m+t-1}{n} = \frac{1}{3}$. This is due to the fact that we are computing the bound specifically for this graph structure, whilst the bounds provided in [6] are independent of the graph structure. If now we consider the 2-distance region \mathcal{C}_2 in figure 2(c), we obtain that $\frac{cc_*}{|\mathcal{C}_2| - nc_*} = \frac{4}{6-0} = \frac{2}{3}$. Again, this leads to a better bound than the one reported in [6] since $\frac{m+t-1}{n} = \frac{1}{2}$. Note that the bounds provided as example are tight. However, despite these examples, the bound assessed by proposition 1 is not guaranteed to be tight and can return worse bounds than the ones provided by the LP-based mechanism.

Both the LP and proposition 1 assess bounds that depend on the graph structure but are independent of the specific reward values. We can always use them to assess bounds independently of the graph structure by assessing the bound for the complete graph, since any other structure is a particular case of the complete graph with some rewards set to zero.

The proof for proposition 1 is a generalization of the one in [9] for k -optimality.

PROOF. For every $C^\alpha \in \mathcal{C}$, consider an assignment x^α such that $x_i^\alpha = x_i^C$ if $x_i \notin C^\alpha$ and $x_i^\alpha = x_i^*$ if $x_i \in C^\alpha$. Since x^C is \mathcal{C} -optimal, for all $C^\alpha \in \mathcal{C}$, $R(x^C) \geq R(x^\alpha)$ holds, and hence

$$R(x^C) \geq \frac{\sum_{C^\alpha \in \mathcal{C}} R(x^\alpha)}{|\mathcal{C}|}. \quad (3)$$

Now for each x^α , we have that $R(x^\alpha) = \sum_{S \in \mathcal{R}} S(x^\alpha)$.

We can split the sum into completely covered ($T(C^\alpha)$), partially covered ($P(C^\alpha)$), or not covered at all ($N(C^\alpha)$) relations, having $R(x^\alpha) = \sum_{S \in T(C^\alpha)} S(x^\alpha) + \sum_{S \in P(C^\alpha)} S(x^\alpha) + \sum_{S \in N(C^\alpha)} S(x^\alpha)$.

Then, by setting partially covered relations to the minimum possible reward (0 assuming non-negative rewards),

$R(x^\alpha) \geq \sum_{S \in T(C^\alpha)} S(x^\alpha) + \sum_{S \in N(C^\alpha)} S(x^\alpha)$. Now, by definition of x^α , for every variable x_i in a relation completely covered by C^α we have that $x_i^\alpha = x_i^*$, and for every variable x_i in a relation not covered at all by C^α we have that $x_i^\alpha = x_i^c$. Hence, $R(x^\alpha) \geq \sum_{S \in T(C^\alpha)} S(x^*) + \sum_{S \in N(C^\alpha)} S(x^c)$. To assess a bound, after substituting this inequality in equation 3, we have that

$$R(x^c) \geq \frac{\sum_{C^\alpha \in \mathcal{C}} \sum_{S \in T(C^\alpha)} S(x^*) + \sum_{C^\alpha \in \mathcal{C}} \sum_{S \in N(C^\alpha)} S(x^c)}{|\mathcal{C}|}. \quad (4)$$

We need to express the numerator in terms of $R(x^c)$ and $R(x^*)$. Grouping the sum by relations and reminding that $cc_* = \min_{S \in \mathcal{R}} cc(S, \mathcal{C})$, the term on the left can be expressed as:

$$\begin{aligned} \sum_{C^\alpha \in \mathcal{C}} \sum_{S \in T(C^\alpha)} S(x^*) &= \sum_{S \in \mathcal{R}} cc(S, \mathcal{C}) \cdot S(x^*) \geq \\ &\geq \sum_{S \in \mathcal{R}} cc_* \cdot S(x^*) = cc_* \sum_{S \in \mathcal{R}} S(x^*) = cc_* \cdot R(x^*). \end{aligned}$$

Furthermore, recalling that $nc_* = \min_{S \in \mathcal{R}} nc(S, \mathcal{C})$, we can do the same with the right term:

$$\begin{aligned} \sum_{C^\alpha \in \mathcal{C}} \sum_{S \in N(C^\alpha)} S(x^c) &= \sum_{S \in \mathcal{R}} nc(S, \mathcal{C}) \cdot S(x^c) \geq \\ &\geq \sum_{S \in \mathcal{R}} nc_* \cdot S(x^c) = nc_* \sum_{S \in \mathcal{R}} S(x^c) = nc_* \cdot R(x^c). \end{aligned}$$

After substituting these two results in equation 3 and rearranging terms, we obtain equation 2. \square

In the next two sections we show that the constant-time reward-independent bounds provided for size and distance optimality in [6, 9] are particular cases of proposition 1.

3.4 Size-optimal bounds as a specific case of region-optimal bounds

Now we present the main result in [9] as a specific case of \mathcal{C} -optimality. An assignment is k -size-optimal if it can not be improved by changing the value of any group of size k or fewer variables.

PROPOSITION 2. *Let $\langle \mathcal{X}, \mathcal{D}, \mathcal{R} \rangle$ be a DCOP with non-negative rewards and m the maximum relation arity. Then, for any k -optimal assignment x^k :*

$$R(x^k) \geq \frac{\binom{|\mathcal{X}|-m}{k-m}}{\binom{|\mathcal{X}|}{k} - \binom{|\mathcal{X}|-m}{k}} R(x^*) \quad (5)$$

PROOF. This result is just a specific case of our general result where we take as region all subsets of size k , that is $\mathcal{C} = \{C^\alpha \subseteq \mathcal{X} \mid |C^\alpha| = k\}$. The number of neighbourhoods is $|\mathcal{C}| = \binom{|\mathcal{X}|}{k}$. The number of neighbourhoods that completely cover S is $cc(S, \mathcal{C}) = \binom{|\mathcal{X}|-|S|}{k-|S|}$, where $|S|$ stands for the cardinality of S (take the variables in S plus $k - |S|$ variables out of the remaining $|\mathcal{X}| - |S|$). Because $cc(S, \mathcal{C})$ reaches the minimum value with the maximum value of $|S|$, $cc_* = \binom{|\mathcal{X}|-m}{k-m}$. The number of neighbourhoods in \mathcal{C} that do not cover S at all is $nc(S, \mathcal{C}) = \binom{|\mathcal{X}|-|S|}{k}$ (take k variables out of the remaining $|\mathcal{X}| - |S|$ variables). Because $nc(S, \mathcal{C})$ reaches the minimum value with the maximum value of $|S|$, $nc_* = \binom{|\mathcal{X}|-m}{k}$. Finally, we obtain equation 5 by using $|\mathcal{X}|$, cc_* and nc_* in equation 2, and simplifying. \square

3.5 Distance-optimal bounds as a specific case of region optimal bounds

Now we present the main result in [6] as a specific case of \mathcal{C} -optimality. First, let us notice that the bound in [6] can be more easily proved if the graph is assumed to be connected. After that, we will see that the bound can be improved in the case that the graph is composed of a set of connected components. Consider a connected DCOP with n variables, minimum constraint arity m , non-negative rewards, and globally optimal assignment x^* . It is easy to see that whenever $m + t - 1 > n$, the length of the shortest path between any two nodes is smaller than t , and hence any t -distance optimal assignment will in fact be globally optimal.

PROPOSITION 3. *Let $\langle \mathcal{X}, \mathcal{D}, \mathcal{R} \rangle$ be a connected DCOP with non-negative rewards. Then, whenever $m + t - 1 \leq n$, we can bound the quality of any t -distance optimal assignment x^t as*

$$R(x^t) \geq \frac{(m+t-1)}{n} R(x^*) \quad (6)$$

PROOF. This result is just a specific case of our general result where we take as region the t -distance neighbourhoods for each variable $x \in \mathcal{X}$, that is $\mathcal{C} = \{\Omega_t(x) \mid x \in \mathcal{X}\}$. The number of neighbourhoods in the region is $|\mathcal{C}| = n$. Next, we show that for every relation S , we have that the number of neighbourhoods in \mathcal{C} that completely cover S , $cc(S, \mathcal{C})$ is at least $m+t-1$. The only variables that do not have S in their t -neighbourhood are those variables that are at distance t or more from every variable in S . If no such variables exist, then $cc(S, \mathcal{C}) = n > m+t-1$. Otherwise, let x' be one of these variables. There is a shortest path connecting x' to its closest variable in S (say x). The path must have length at least t , that is $x, x_1, \dots, x_{t-1}, \dots, x'$. Now, it is clear that S is in the t -neighbourhood of the $t-1$ variables $\{x_1, \dots, x_{t-1}\}$. Note that since we are taking the shortest path to any variable in S , no x_i can be in S . Since S is also in the t -neighbourhood of every variable in S and there can be no intersection between S and $\{x_1, \dots, x_{t-1}\}$, we have $cc(S, \mathcal{C}) = |S| + t - 1 \geq m + t - 1$. Hence $cc_* \geq m + t - 1$. By definition, $nc_* \geq 0$. Finally, we obtain equation 6 by using $|\mathcal{C}|$, cc_* and nc_* in equation 2, and simplifying. \square

If the DCOP is not connected, we can obtain a better bound by simply applying equation 6 to each connected component and taking the minimum. That is $R(x^t) \geq \frac{(m+t-1)}{n_*} R(x^*)$ where n_* is the number of elements of the largest connected component, which is always smaller than n .

4. EMPIRICAL EVALUATION

In this section we show how we can benefit from the larger space of criteria for defining regions provided by \mathcal{C} -optimality. We start by analyzing the regions generated by k -size and t -distance on DCOPs with different structures, to conclude that k -size generates a potentially huge number of neighborhoods of limited size and t -distance generates a limited number of potentially huge neighborhoods. To keep under control the amount and size of neighborhoods we introduce a new type of regions, namely size-bounded distance regions, that include a limited number of limited size neighborhoods. Finally, we empirically show that algorithms for approximate DCOP solving can benefit from using size-bounded distance regions.

We start by analyzing k -size and t -distance regions in section 4.1, to motivate the introduction of size-bounded distance regions in section 4.2. The DALO algorithm was proposed in [6] to find either k - or t - optimal solutions. In section 4.3 we show how we can extend it to find an optimal in any region \mathcal{C} . Finally, in section 4.4 we compare the performance of size, distance and size-bounded distance regions on DCOPs with different graph structures using DALO.

4.1 Analysis of size and distance regions

We are interested in analyzing the regions generated by k -size and t -distance on DCOPs with different structures. More concretely, we want to assess the number of different neighbourhoods as well as the size (number of variables) for each neighbourhood, since both parameters strongly influence the amount of computation needed to obtain a k , t -optimum. The worst case time for checking optimality in a neighbourhood is exponential in its number of variables. Furthermore, if an agent has to consider a large number of neighbourhoods, it will have to share its time among them. Hence, in terms of computational effort, it is of interest to find regions that have a limited number of neighbourhoods of limited size. In k -size optimality the size is limited by k but the number of neighbourhoods grows as $\binom{|\mathcal{X}|}{k}$, which can turn out prohibitively large. In t -distance optimality the number of neighbourhoods is $\mathcal{O}(|\mathcal{X}|)$ but the size of the neighbourhoods is not limited. For example, the 1-distance region of a complete graph contains a single neighbourhood with all the variables, and hence finding a 1-distance optimal in a complete graph is as hard as finding a global optimum.

For a more detailed empirical analysis, we have computed statistics of the maximum neighbourhood size in a region (MaxS) and the number of neighbourhoods per agent (#) over randomly generated constraint graphs. We have used three different types of graph structures: $G(n, M)$ random graphs [2], Barabasi-Albert (BA) scale-free graphs [1], and non-linear preferential attachment (NLPA) graphs based on the BA model, but with a larger emphasis on many nodes having fewer connections. All the graphs have 100 nodes with a density of four meaning that on average each node has four neighbours. We compare the results of three different criteria: 5-size (K5)², 1-distance (T1) and 2-distance (T2). The first three rows in table 1 present the averages over 50 DCOPs of MaxS and # for each criteria and each type of graphs. From these statistics we observe that T1 and T2 distance criteria result in very large neighbourhoods, especially on scale-free and NLPA graphs due to the presence of *hub* agents with a large number of neighbours. We also observe that K5 criterion generates a large number of neighbourhoods, specially in scale-free and NLPA due to the presence of hub nodes (e.g. the average number of neighbourhoods per agent in NLPA graphs is 11366).

From this analysis we can conclude that k -size generates a potentially huge number of neighborhoods of limited size and t -distance generates a limited number of potentially huge neighborhoods. To overcome this, we introduce a new type of regions, namely size-bounded distance regions, which include a limited number of bounded size neighborhoods.

4.2 Size-bounded distance optimality

Our aim at formulating the size-bounded distance crite-

²As in [6, 9] neighbourhoods of 5 variables that are not connected in the graph are discarded.

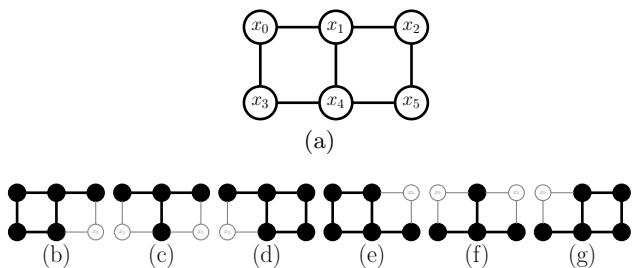


Figure 3: Example of (a) a DCOP graph, and (b)-(g) the set neighborhoods for the 5-size-distance bounded region.

tion is to provide an alternative trade-off to size and distance, being more aware of the complexity of the regions they generate.

Let $T(x_i, x_j)$ be the distance between two variables in the constraint graph. Let $\Omega_t(x_i) = \{x_j | T(x_i, x_j) \leq t\}$ be the t -distance neighbourhood centered on variable x_i . Then, the s -size-bounded-distance neighbourhood is the largest t -distance region whose number of variables does not exceed the limit s . Formally, let $t(x_i) = \max \{t \text{ s.t. } |\Omega_t(x_i)| \leq s\}$ be the largest value for t such that $|\Omega_t(x_i)| \leq s$. The s -size-bounded-distance neighbourhood centered on variable x_i is defined as $\Phi_s(x_i) = \Omega_{t(x_i)}(x_i)$.

Figure 3 (b)-(g) depicts 5-size-bounded distance neighbourhoods for agents x_0 to x_5 for the DCOP in figure 3 (a). Observe that agents can end up exploring different distance levels in their neighbourhoods as a result of bounding their size to s . In our example, agents x_0, x_2, x_3 and x_5 explore their 2-distance neighbourhood with size 5 (figures 3 (b)(d)(e)(g)), whereas agents x_1 and x_4 are restricted to 1-distance neighbourhood with size 4 (figures 3 (c)(f)).

Now, the s -size-bounded distance region includes the s -size-bounded-distance neighbourhood of each agent $x_i \in \mathcal{X}$. Moreover, in order to ensure that all relations are covered, the s -size-bounded-distance region also includes a neighbourhood for every edge in the graph.

Note that in size-bounded distance optimality both the number of neighbourhoods and their size are limited. Now we can go back to table 4, to compare the number of regions and its size with the state-of-the-art criteria. In the last row we show the averages over 50 constraint graphs of MaxS and # for 5-size-bounded-distance optimality (S5) for each type of graph. We can see that S5 is the only criterion that manages to keep the size of the region limited (to 5 agents) together with a reasonable number of neighbourhoods per agent (between 3 and 10 depending on the graph structure).

4.3 DALO for region optimality

The DALO algorithm is an asynchronous algorithm that starts with a random initial assignment and monotonically increases the solution quality by independently optimizing in each of the neighbourhoods that are created. As described in [6], DALO has three phases: initialization, optimization, and implementation.

During the initialization phase agents distributedly create a set of neighbourhoods and assign each neighbourhood to a leader agent (the central node to minimize communication) that will be in charge of its optimization. After initialization, agents run in parallel the optimization and implementation phases for each assigned neighbourhood until stabilization. During the optimization phase, each leader agent optimises by searching for a joint assignment of the variables in its

neighborhoods that improve their reward. After optimizing, the leader agent runs the implementation phase trying to implement the new joint assignment found. Because neighbourhoods are optimised in parallel and a variable can appear in multiple neighbourhoods, DALO implementation phase uses an asynchronous protocol based on a standard lock/commit pattern to ensure stability.

To use DALO with an arbitrary region, we focused on the initialization phase to modify how agents create the groups over which they optimise. Concretely to allow DALO to search for a \mathcal{C} -optimal, agents will distributedly generate the neighbourhoods in region \mathcal{C} . For example, to use DALO in the s -size-bounded distance region each agent will iterate through various t -distance neighbourhoods, by broadcasting at distance t , to determine the largest t -distance neighbourhood whose size does not exceed s . After initialization, for the specific region, optimization and implementation phases are ran as specified in [6], independently of the used region.

4.4 Empirical results

In this section we compare the results obtained by DALO using four different criteria: 5-size ($K5$), 1-distance ($T1$), 2-distance ($T2$), and 5-size bounded distance ($S5$) criteria.

We ran similar experimental settings to Kiekintveld et al. [6]. We measured the performance of the extension of the DALO algorithm³ described in section 4.3 when running over each one of the regions generated by the four criteria described above. Thus, we tested DALO for the four criteria over the different types of graphs described in section 4.1. All the graphs have 100 nodes, each one with density 4, meaning that on average each node has 4 neighbours. Moreover, variables' domain size is 10, and rewards are integers sampled from a distribution $U[0, 10000]$.

Besides graph types, we also considered different *Computation/Communication Ratios* (CCR) [6]. The CCR setting defines the number of constraint assignments that may be evaluated at each communication step. For example, $CCR = 0.01$ allows each node to process up to 100 checks in a time step. We vary the setting of CCR in our experiments to test DALO in two settings with different relative cost for sending messages and computation, namely $CCR = 0.01$ and $CCR = 0.1$. In general, the larger the value of CCR, the higher the computation cost. Notice that, with respect to the experimental settings in [6], we discarded using $CCR = 0$. The rationale is that if $CCR = 0$, communication is infinitely more costly than computation and hence the best strategy is computing the optimal by means of a fully centralized algorithm.

Figures 4 (a)-(f) plot the normalized solution quality of each algorithm along global time for each graph structure and CCR metric. The normalized solution quality is computed by: (1) subtracting the initial reward, as assessed by DALO for a given criterion, from the reward at a given global time; and (2) dividing the result by the best known reward obtained by DALO out of the four criteria. All results are averaged over 25 sample instances. In what follows, we compare the four criteria along two dimensions: (1) the final normalised solution quality; and (2) the convergence speed required to reach a good solution quality.

Regarding solution quality, the results vary depending the value of CCR and graph structure. On the one hand, in

	Random		Scale-free		NPLA	
	MaxS	#	MaxS	#	MaxS	#
K5	5	167	5	963	5	11366
T1	10	1	27	1	63	1
T2	38	1	82	1	99	1
S5	5	3	5	3	5	10

Table 1: Statistics for regions generated by k5, t1, t2 and s5 criteria for 100 agents. MaxS stands for the maximum size of a neighbourhood and# for the average number of neighbourhoods per agent.

scenarios where computation is more costly ($CCR = 0.1$), overall $S5$ outperforms the rest of criteria. Although $T1$ is very competitive and its solution quality comes very close to that of $S5$ over random and scale free graphs, $S5$ significantly outperforms $T1$ on NPLA graphs. Moreover, both $S5$ and $T1$ largely outperform $K5$. The reason of the poor performance of $K5$ is that it generated neighbourhoods of fixed size. On the other hand, in scenarios where computation is cheaper ($CCR = 0.01$), the differences of final solution qualities between $S5$, $T1$, and $K5$ are not significant. There is an aspect though that deserves special attention. Notice that for all the test cases, the performance of DALO over $T2$ regions is much worse than the performance over the regions generated by the rest of criteria. We can explain this result by analysing the complexity of $T2$ regions as shown in table 1. Thus, we observe that $T2$ generates very large neighbourhoods that can not be optimised within the maximum global time (1000 global time steps). The solution quality degradation when handling $T2$ regions is particularly significant on scale-free and NPLA graphs because the criterion generates neighbourhoods whose size is close to the size of the original problem (99 variables on average in NPLA graphs).

Regarding convergence speed, $S5$ regions help DALO converge to a high solution quality faster. Likewise our analysis about solution quality above, $T1$ is again competitive with respect to $S5$, though $S5$ largely outperforms $T1$ on NPLA graphs. This is because, as observed in [6], NPLA graphs are characterized by large *hub* nodes with many connections that results in large neighbourhoods that take long for agents to optimise. Regarding $K5$, convergence speed is slower than that of $S5$ and $T1$ because each leader in DALO coordinates a neighbourhood of size 5, whereas the neighbourhoods for $S5$ and $T1$ may be smaller.

To summarise, our experimental results show that criteria that produce regions with large number of neighbourhoods or/and large neighbourhood sizes are not guaranteed to outperform criteria that produce less complex regions. In fact, overall the size-bounded distance criterion proposed in section 4.2 was able to outperform the rest of criteria by limiting the complexity of the regions that it generates.

5. CONCLUSIONS

In this paper we generalise the k - and t -optimal frameworks [9, 6] to introduce \mathcal{C} -optimality, a flexible framework that provides quality guarantees for local optima in regions characterised by any arbitrary criterion. With this aim, we provide: (1) a formal definition of \mathcal{C} -optimality, namely of local optimality in some arbitrary region; and (2) quality guarantees for region optimal solutions that exploit the knowledge about the graph structure. Regarding quality guarantees, we defined two methods with different computational costs: (1) a first one, based on solving an LP, that guarantees tightness; and (2) a second one that requires lin-

³We used the DALO code provided by the authors at <http://teamcore.usc.edu/dcop/>.

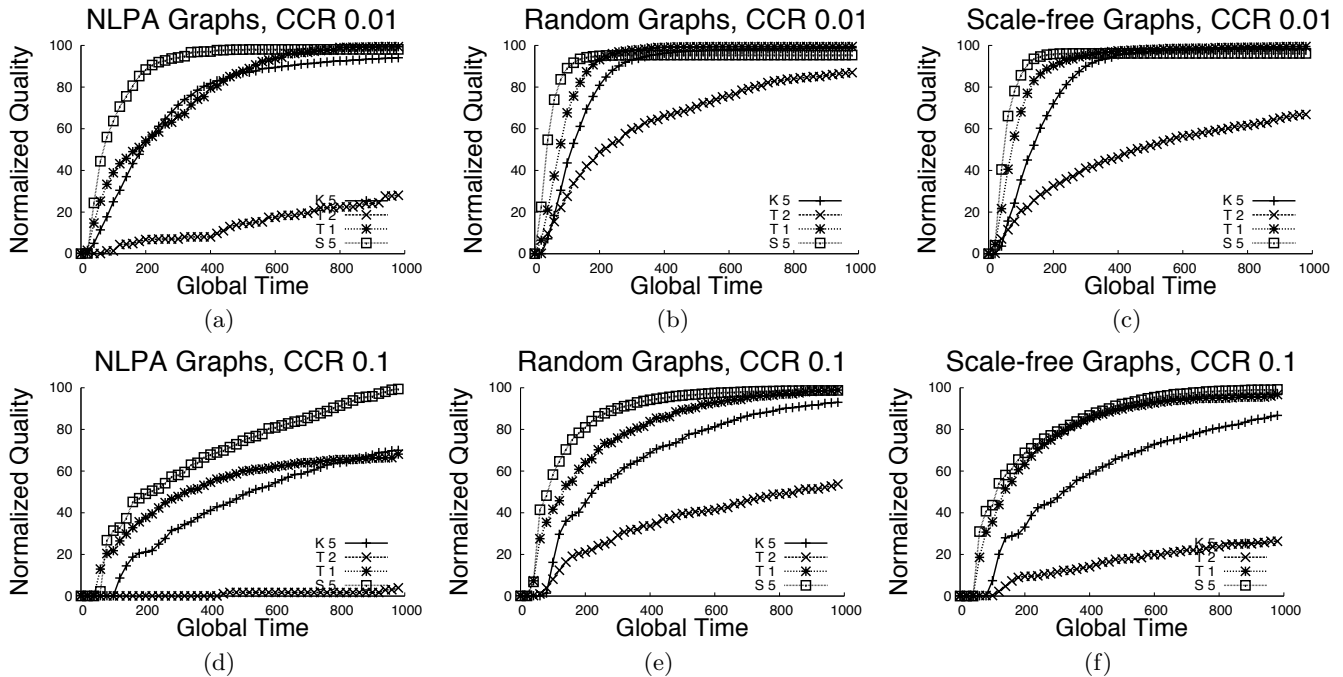


Figure 4: Experimental results comparing DALO for $K5$, $T1$, $T2$ and $S5$ regions.

ear time but does not ensure tightness. Moreover, we prove that the bounds provided for size and distance optimality are particular instances of the C -optimal bounds.

To illustrate how the C -optimality framework allows us to explore the space for arbitrary criteria, we proposed a novel criterion to generate regions, the so-called size-bounded-distance criterion. This new criterion has been designed to overcome the main drawbacks of size and distance optimality. Moreover, we extend the DALO algorithm [6] to compute C -optimal solutions. Our empirical analysis of the size-bounded-distance criterion shows that it outperforms both size and distance criteria by providing a more fine-grained control of the complexity of the regions to explore.

As future work, we plan to extend the C -optimal bounds to exploit some a-priori knowledge of the reward structure of the problem, if available, along the lines of [3]. Furthermore, since a critical issue in the design of any C -optimal algorithm is the choice of regions, we will focus on defining techniques that allow us to explore the space of regions in search for regions with limited complexity and high quality guarantees.

Acknowledgments. Members of the Teamcore Research Group thanks Perceptronic Inc for their support of the research. Work funded by projects EVE (TIN2009-14702-C02-01 and 02), AT(CONSOLIDER CSD2007-0022), and Generalitat de Catalunya (2009-SGR-1434). Vinyals is supported by the Ministry of Education of Spain (FPU grant AP2006-04636).

6. REFERENCES

- [1] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.
- [2] B. Bollobas. *Random Graphs*. Cambridge Press, 2001.
- [3] E. Bowring, J. P. Pearce, C. Portway, M. Jain, and M. Tambe. On k -optimal distributed constraint optimization algorithms: new bounds and algorithms. In *AAMAS (2)*, pages 607–614, 2008.
- [4] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *AAMAS (2)*, pages 639–646, 2008.
- [5] R. Junges and A. L. C. Bazzan. Evaluating the performance of DCOP algorithms in a real world, dynamic problem. In *AAMAS*, pages 599–606, 2008.
- [6] C. Kiekintveld, Z. Yin, A. Kumar, and M. Tambe. Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In *AAMAS*, pages 133–140, 2010.
- [7] R. Mailler and V. R. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *AAMAS*, pages 438–445, 2004.
- [8] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artif. Intell.*, 161(1-2):149–180, 2005.
- [9] J. P. Pearce and M. Tambe. Quality guarantees on k -optimal solutions for distributed constraint optimization problems. In *IJCAI*, pages 1446–1451, 2007.
- [10] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, pages 266–271, 2005.
- [11] M. Vinyals, J. Cerquides, A. Farinelli, and J. A. Rodriguez-Aguilar. Worst-case bounds on the quality of max-product fixed-points. In *NIPS*, December 2010.
- [12] M. Vinyals, M. Pujol, J. Rodriguez-Aguilar, and J. Cerquides. Divide and Coordinate: solving DCOPs by agreement. In *AAMAS*, pages 150–156, 2010.
- [13] M. Vinyals, E. Shieh, J. Cerquides, J. A. Rodriguez-Aguilar, Z. Yin, M. Tambe, and E. Bowring. LP formulation of regional-optimal bounds. Technical Report TR-III-A-2011-01, 2011. At: <http://www.iiia.csic.es/files/pdfs/TR201101.pdf>.
- [14] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artif. Intell.*, 161(1-2):55–87, 2005.

Distributed Algorithms for Solving the Multiagent Temporal Decoupling Problem

James C. Boerkoel Jr. and Edmund H. Durfee
Computer Science and Engineering
University of Michigan, Ann Arbor, MI 48109, USA
{boerkoel, durfee}@umich.edu

ABSTRACT

Scheduling agents can use the Multiagent Simple Temporal Problem (MaSTP) formulation to efficiently find and represent the complete set of alternative consistent joint schedules in a distributed and privacy-maintaining manner. However, continually revising this set of consistent joint schedules as new constraints arise may not be a viable option in environments where communication is uncertain, costly, or otherwise problematic. As an alternative, agents can find and represent a temporal decoupling in terms of locally independent sets of consistent schedules that, when combined, form a set of consistent joint schedules. Unlike current algorithms for calculating a temporal decoupling that require centralization of the problem representation, in this paper we present a new, provably correct, distributed algorithm for calculating a temporal decoupling. We prove that this algorithm has the same theoretical computational complexity as current state-of-the-art MaSTP solution algorithms, and empirically demonstrate that it is more efficient in practice. We also introduce and perform an empirical cost/benefit analysis of new techniques and heuristics for selecting a maximally flexible temporal decoupling.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence

General Terms

Algorithms, Experimentation, Theory

Keywords

Multiagent Scheduling, Temporal Decoupling Problem

1. INTRODUCTION

A scheduling agent is often responsible for independently managing the scheduling constraints of its user, while also ensuring that its user's schedule coordinates with the schedules of other agents' users. In many scheduling environments, agents must also react to new constraints that arise over time, either due to volitional decisions by the agents (or their

Cite as: Distributed Algorithms for Solving the Multiagent Temporal Decoupling Problem, James C. Boerkoel Jr. and Edmund H. Durfee, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 141-148.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

users) or due to the dynamics of the environment. Agent coordination is further challenged by desires for privacy and uncertain, costly, or otherwise problematic communication. Fortunately, scheduling agents can use the Multiagent Simple Temporal Problem (MaSTP) formulation to, in a distributed and efficient manner, find and represent sets of alternative consistent joint schedules to these types of complex, multiagent scheduling problems [3, 2].

As an example of this type of problem, suppose three student colleagues, Ann, Bill, and Chris, have each selected a tentative morning schedule (from 8:00 to noon) and have each tasked a personal computational scheduling agent with maintaining his/her schedule. Ann will have a 60 minute run with Bill before spending 90 to 120 minutes on a group project (after picking up deliverables that Chris will leave in the lab); Bill will have a 60 minute run with Ann before spending 60 to 180 minutes working on homework; and finally, Chris will work on the group project for 90-120 minutes and drop it off in the lab before attending a lecture from 10:00 to 12:00. This example is displayed graphically as a distance graph (explained in Section 2.1) in Figure 1(a).

One approach for solving this problem is to represent the set of *all* possible joint schedules that satisfy the constraints, as displayed in Figure 1(b). In this approach, if a new, non-volitional constraint arrives (e.g., Chris' bus is late), the agents can easily recover by simply eliminating inconsistent joint schedules from consideration. However, doing so may still require communication (e.g., Chris' agent should communicate that her late start will impact when Ann can start, and so on). In fact, this communication must continue (e.g., until Chris actually completes the project, Ann does not know when she can start), otherwise agents could make inconsistent, concurrent decisions. For example, if Ann decides she wants to run at 8:00, while Bill simultaneously decides he wants to run at 9:00 (both allowable possibilities), Ann and Bill's agents will inadvertently introduce an inconsistency. An alternative to this approach, displayed graphically in Figure 1(d), is for agents to simply select one joint schedule from the set of possible solutions. However, as soon as a new, non-volitional constraint arrives (e.g., Chris' bus arrives late by even a single minute), this exact, joint solution may no longer be valid. This, in turn, can require agents to regenerate a new solution *every* time a new constraint arrives, unless that new constraint is consistent with the selected schedule. Due to either a lack of robustness or lack of independence, neither of these two approaches is likely to perform well in time-critical, highly-dynamic environments.

Fortunately, there is a third approach that balances the

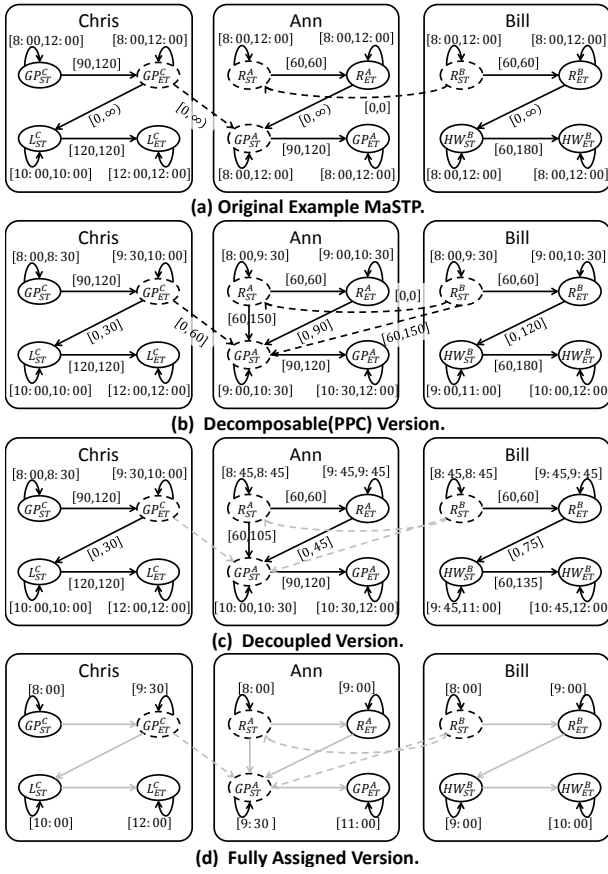


Figure 1: The distance graph corresponding to the (a) original, (b) decomposable, (c) decoupled, and (d) fully assigned, versions of our example MaSTP.

robustness of Figure 1(b) with the independence of Figure 1(d). Agents can find and maintain a temporal decoupling, which is composed of *independent* sets of locally consistent schedules that, when combined, form a set of consistent joint schedules [4]. An example of a temporal decoupling is displayed in Figure 1(c), where, for example, Chris’ agent has agreed to complete the group project by 10:00 and Ann’s agent has agreed to wait to begin work until after 10:00. Not only are agents’ schedules no longer interdependent, but agents also still maintain sets of locally consistent schedules. Now when Chris’ bus is late by a minute, Chris’ agent can “absorb” this new constraint by independently updating its local set of schedules, without requiring any communication with any other agent. The advantage of this approach is that once agents establish a temporal decoupling, there is no need for further communication unless (or until) a new (series of) non-volitional constraint(s) render the chosen decoupling inconsistent. It is only if and when a temporal decoupling does become inconsistent (e.g., Chris’ bus is more than a half hour late, violating her commitment to finish the project by 10:00) that agents must calculate a new temporal decoupling (perhaps establishing a new hand-off deadline of 10:15), and then once again independently react to newly-arriving constraints, repeating the process as necessary.

Unfortunately, the current temporal decoupling algorithms [4, 7] require centralizing the problem representation at some

“coordinator” who sets the decoupling constraints for all. The computational, communication, and privacy costs associated with centralization may be unacceptable in multiagent planning and scheduling applications, such as military, health care, or disaster relief, where agents specify problems in a distributed fashion, expect some degree of privacy, and must provide unilateral, time-critical, and coordinated scheduling assistance. In this paper, we contribute new, *distributed* algorithms for calculating a temporal decoupling, prove the correctness, privacy implications, and runtime properties of these algorithms, and perform an empirical comparison that shows that these algorithms calculate a temporal decoupling that approaches the best centralized methods in terms of flexibility, but with less computational effort than current MaSTP solution algorithms.

2. PRELIMINARIES

In this section we provide definitions necessary for understanding our contributions, using and extending terminology from the literature.

2.1 Simple Temporal Problem

As defined in [3], the Simple Temporal Problem (STP), $\mathcal{S} = \langle V, C \rangle$, consists of a set of timepoint variables, V , and a set of temporal difference constraints, C . Each timepoint variable represents an event, and has an implicit, continuous numeric domain. Each temporal difference constraint c_{ij} is of the form $v_j - v_i \leq b_{ij}$, where v_i and v_j are distinct timepoints, and $b_{ij} \in \mathbb{R}$ is a real number bound on the difference between v_j and v_i . To exploit extant graphical algorithms and efficiently reason over the constraints of an STP, each STP is associated with a weighted, directed graph, $\mathcal{G} = \langle V, E \rangle$, called a *distance graph*. The set of vertices V is as defined before (each timepoint variable acts as a vertex in the distance graph) and E is a set of directed edges, where, for each constraint c_{ij} of the form $v_j - v_i \leq b_{ij}$, we construct a directed edge, e_{ij} from v_i to v_j with an initial weight $w_{ij} = b_{ij}$. As a graphical short-hand, each edge from v_i to v_j is assumed to be bi-directional, compactly capturing both edge weights with a single label, $[-w_{ji}, w_{ij}]$, where $v_j - v_i \in [-w_{ji}, w_{ij}]$ and w_{ij} is initialized to ∞ if there exists no corresponding constraint c_{ij} in P . All times (e.g. ‘clock’ times) can be expressed relative to a special *zero* timepoint variable, $z \in V$, that represents the “start of time”. Bounds on the difference between v_i and z are expressed graphically as “unary” constraints specified over a timepoint variable v_i . Moreover, w_{zi} and w_{iz} then represent the earliest and latest times, respectively, that can be assigned to v_i , and thus implicitly define v_i ’s domain. In this paper, we will assume that z is always included in V and that, during the construction of \mathcal{G} , an edge e_{zi} is added from z to every other timepoint variable $v_i \in V$.

An STP is *consistent* if there exist no negative cycles in the corresponding distance graph. A consistent STP contains at least one *solution*, which is an assignment of specific time values to timepoint variables that respects all constraints to form a *schedule*. A *decomposable* STP represents the entire set of solutions by establishing the tightest bounds on timepoint variables such that: (1) no solutions are eliminated and (2) any assignment of a specific time to a timepoint variable that respects these bounds can be extended to a solution with a backtrack-free search using constraint propagation. *Full-Path Consistency* (FPC) works by establishing de-

composability of an STP instance in $O(|V|^3)$ by applying an all-pairs-shortest-path algorithm, such as Floyd-Warshall, to the distance graph to find the tightest possible path between every pair of timepoints, v_i and v_j , forming a fully-connected graph, where $\forall i, j, k, w_{ij} \leq w_{ik} + w_{kj}$. The resulting graph is then checked for consistency by validating that there are no negative cycles, that is, $\forall i \neq j$, ensuring $w_{ij} + w_{ji} \geq 0$ [3].

An alternative for checking STP consistency is to establish **Directed Path Consistency** (DPC) [3] on its distance graph. DPC *triangulates* the distance graph by visiting each timepoint, v_k , in some **elimination order**, $o = (v_1, v_2, \dots, v_n)$, tightening (and when necessary, adding) edges between each pair of its not yet eliminated neighboring timepoints, v_i, v_j (connected to v_k via an edge), using the rule $w_{ij} \leftarrow \min(w_{ij}, w_{ik} + w_{kj})$, and then “eliminating” that timepoint from further consideration. The quantity ω_o^* is the *induced graph width* relative to o , and is defined as the maximum, over all v_k , of the size of v_k ’s set of not yet eliminated neighbors at the time of its elimination. The edges added during this process, along with the existing edges, form a **triangulated** (also called chordal) **graph** — a graph whose largest non-bisected cycle is of size three. The complexity of DPC is $O(|V| \cdot \omega_o^{*2})$, but instead of establishing decomposability, it establishes the property a solution can be recovered from a DPC distance graph in a backtrack-free manner if variables are assigned in reverse elimination order. **Partial Path Consistency** (PPC) [1] is sufficient for establishing decomposability on an STP instance by calculating the tightest possible path for *only* the subset of edges that exists within a triangulated distance graph. As a result, PPC may establish decomposability much faster than FPC algorithms in practice ($O(|V| \cdot \omega_o^{*2}) \subseteq O(|V|^3)$) [8, 6]. The PPC representation of our example is displayed in Figure 1(b).

2.2 Multiagent Simple Temporal Problem

The Multiagent Simple Temporal Problem (MaSTP) is informally composed of n local STP subproblems, one for each of n agents, and a set of constraints C_X that establish relationships between the local subproblems of different agents. Our definition of the MaSTP improves on our original MaSTP specification [2]. An agent i ’s **local** STP subproblem is defined as $\mathcal{S}_L^i = \langle V_L^i, C_L^i \rangle^1$, where:

- V_L^i is defined as agent i ’s set of **local variables**, which is composed of all timepoints *assignable* by agent i and also includes agent i ’s reference to z ;
- C_L^i is defined as agent i ’s set of intra-agent or **local constraints**, where a local constraint, $c_{ij} \in C_L^i$ is defined as a bound on the difference between two local variables, $v_j - v_i \leq b_{ij}$, where $v_i, v_j \in V_L^i$.

In Figure 1(a), the boxes labeled Chris, Ann, and Bill represent each person’s respective local STP subproblem from our running example. Notice, the sets V_L^i partition the set of all non-reference timepoint variables and the sets C_L^i partition the set of all local constraints.

Moreover, C_X is the set of inter-agent or **external constraints**, where an external constraint is defined as a bound on the difference between two variables that are local to different agents, $v_i \in V_L^i$ and $v_j \in V_L^j$, where $i \neq j$. Further, V_X is defined as the set of **external timepoint variables**,

¹Throughout this paper we will use superscripts to index agents and subscripts to index variables and edges.

where a timepoint is external if it is involved in at least one external constraint. In Figure 1(a), external constraints and variables are denoted with dashed edges. It then follows that:

- C_X^i is agent i ’s set of external constraints that each involve exactly one of agent i ’s assignable timepoints;
- V_X^i is the set of timepoint variables known to agent i due their involvement in some constraint from C_X^i , but that are local to some other agent $j \neq i$.

More formally, then, an MaSTP, \mathcal{M} , is defined as the STP $\mathcal{M} = \langle V_{\mathcal{M}}, C_{\mathcal{M}} \rangle$ where $V_{\mathcal{M}} = \{\bigcup_i V_L^i\}$ and $C_{\mathcal{M}} = \{C_X \cup \bigcup_i C_L^i\}$. Note, the definition of the corresponding distance graph is defined as before, where the definition of agent i ’s local and external edges, E_L^i and E_X^i , follows analogously from the definition of C_L^i and C_X^i , respectively.

2.3 Multiagent Temporal Decoupling Problem

Given the previous definitions, we adapt the definition of temporal decoupling in [4] to apply to the MaSTP. Agents’ **local STP subproblems** $\{\mathcal{S}_L^1, \mathcal{S}_L^2, \dots, \mathcal{S}_L^n\}$ form a **temporal decoupling** of an MaSTP \mathcal{M} if:

- $\{\mathcal{S}_L^1, \mathcal{S}_L^2, \dots, \mathcal{S}_L^n\}$ are consistent STPs; and
- Merging *any* locally consistent solutions to the problems in $\{\mathcal{S}_L^1, \mathcal{S}_L^2, \dots, \mathcal{S}_L^n\}$ yields a solution to \mathcal{M} .

Alternatively, when $\{\mathcal{S}_L^1, \mathcal{S}_L^2, \dots, \mathcal{S}_L^n\}$ form a temporal decoupling of \mathcal{M} , $\{\mathcal{S}_L^1, \mathcal{S}_L^2, \dots, \mathcal{S}_L^n\}$ are said to be **temporally independent**. The Multiagent Temporal Decoupling Problem (MaTDP), then, is defined as, for each agent i , finding a set of constraints C_{Δ}^i such that if $\mathcal{S}_{L+\Delta}^i = \langle V_L^i, C_N^i \cup C_{\Delta}^i \rangle$, then $\{\mathcal{S}_{L+\Delta}^1, \mathcal{S}_{L+\Delta}^2, \dots, \mathcal{S}_{L+\Delta}^n\}$ is a temporal decoupling of MaSTP \mathcal{M} . Figure 1(c) represents a temporal decoupling of our example, where new unary decoupling constraints, in essence, replace all external edges (shown faded). A **minimal decoupling** is one where, if the bound of any decoupling constraint $c \in C_{\Delta}^i$ for some agent i is relaxed, then $\{\mathcal{S}_{L+\Delta}^1, \mathcal{S}_{L+\Delta}^2, \dots, \mathcal{S}_{L+\Delta}^n\}$ is no longer a decoupling (e.g., Figure 1(c) is an example of a minimal decoupling whereas the decoupling in (d) is not minimal). The original TDP algorithm [4] executes on a centralized representation of the MaSTP and iterates between proposing new constraints to decouple agent subproblems with respect to a particular external constraint (until all external constraints have been decoupled) and reestablishing FPC on the corresponding global distance graph, so that subsequently proposed decoupling constraints are guaranteed to be consistent.

3. ALGORITHMS

In this section, we introduce new *distributed* algorithms for calculating a temporal decoupling and prove their correctness, computational complexity, and privacy properties.

3.1 A Distributed MaTDP Algorithm

The goal of our Multiagent Temporal Decoupling Problem (MaTDP) algorithm, presented as Algorithm 1, is to find a set of decoupling constraints C_{Δ} that render the external constraints C_X moot. Agents accomplish this goal by coordinating both to establish DPC and also to consistently assign external variables in reverse elimination order. First, we use D Δ P3C-1 (an efficient, distributed DPC algorithm corresponding to lines 1-22 of the D Δ P3C algorithm presented in [2]) to triangulate and propagate the constraints,

Algorithm 1 Multiagent Temporal Decoupling Problem (MaTDP) Algorithm

Input: \mathcal{G}^i , agent i 's known portion of the distance graph corresponding an MaSTP instance \mathcal{M} .

Output: C_Δ^i , agent i 's decoupling constraints, and \mathcal{G}^i , agent i 's PPC distance graph w.r.t. C_Δ^i .

- 1: $\mathcal{G}^i, o_L^i, o_X = (v_1, v_2, \dots, v_n) \leftarrow \text{D}\Delta\text{P3C-1}(\mathcal{G}^i)$
 - 2: Return INCONSISTENT if $\text{D}\Delta\text{P3C-1}$ does
 - 3: $C_\Delta^i = \emptyset$
 - 4: **for** $k = n \dots 1$ such that $v_k \in V_L^i$ **do**
 - 5: $w_{zk}^{DPC} \leftarrow w_{zk}, w_{kj}^{DPC} \leftarrow w_{kj}$
 - 6: **for** $j = n \dots k + 1$ such that $\exists e_{jk} \in E_L^i \cup E_X^i$ **do**
 - 7: **if** $e_{jk} \in E_X^i$ **then**
 - 8: $w_{zj}, w_{jz} \leftarrow$ Block until receive updates from ($\text{Agent}(v_j)$)
 - 9: **end if**
 - 10: $w_{zk} \leftarrow \min(w_{zk}, w_{zj} + w_{jk})$
 - 11: $w_{kj} \leftarrow \min(w_{kj}, w_{kj} + w_{jz})$
 - 12: **end for**
 - 13: Assign v_k // tighten w_{zk}, w_{kj} to ensure $w_{zk} + w_{kj} = 0$
 - 14: Send w_{zk}, w_{kj} to each $\text{Agent}(v_j)$ s.t. $j < k, e_{jk} \in E_X^i$
 - 15: $C_\Delta^i \leftarrow C_\Delta^i \cup \{(z - v_k \in [-w_{zk}, w_{kj}])\}$
 - 16: **end for**
 - 17: **if** (RELAX) **then** $\mathcal{G}^i, C_\Delta^i \leftarrow \text{MaTDR}(\mathcal{G}^i, w^{DPC})$
 - 18: **return** P3C-2($\mathcal{G}_{L+\Delta}^i, o_L^i$), C_Δ^i
-

where external timepoints V_X are eliminated last. Figure 2(a) shows V_X after all other local variables have been eliminated. Notice that local constraints are reflected in the tighter domains. The external variables are eliminated in order, from left to right ($o_X = (GP_{ET}^C, R_{ST}^A, GP_{ST}^A, R_{ST}^B)$), which introduces the new edges, shown with dotted lines, and their weights. If $\text{D}\Delta\text{P3C-1}$ propagates to an inconsistent graph, then our algorithm returns INCONSISTENT.

Otherwise, we initialize an empty C_Δ and then step through vertices in inverse elimination order, starting with R_{ST}^B . We skip over the inner loop (lines 6-12) because there are no vertices later in o_X than R_{ST}^B . In line 13, we use a heuristic that decouples by ‘‘assigning’’ the timepoint to the midway point between its upper and lower bounds. In this case we add the constraint that R_{ST}^B happens at 8:45 to C_Δ (line 15). In line 14, this is sent to Ann’s agent, because R_{ST}^B shares external edges with Ann’s timepoints. The next vertex is GP_{ST}^A . Note, Ann’s agent would consider processing this variable right away, but the inner loop (lines 6-12) forces Ann’s agent to wait for the message from Bill’s agent. When it gets there, Ann’s agent updates its edge weights accordingly (lines 10-11). In this case, given that GP_{ST}^A is at least 60 minutes after R_{ST}^B , GP_{ST}^A ’s domain is tightened to [9:45, 10:30]. Then in line 13, Ann’s agent chooses the decoupling point by splitting the difference, thus adding the constraint that G_{ST}^A occurs at 10:08. This same process is repeated until all timepoints in V_X have been assigned; the result is shown in Figure 2.

As mentioned, our default heuristic is to assign v_k to the midpoint of its path consistent domain (which corresponds to using the rules $w_{zk} \leftarrow w_{zk} - \frac{1}{2}(w_{zk} + w_{kj}); w_{kj} \leftarrow -w_{zk}$ for line 13). In general, however, assigning variables is more constraining than necessary. Fortunately, agents can optionally call a *relaxation* algorithm (introduced in Section 3.2) that replaces C_Δ with a set of *minimal* decoupling constraints. Later in this paper, we will explore and evaluate other assignment heuristics for line 13 (other than our default midpoint assignment procedure) that, when combined with the relaxation algorithm, could lead to less constraining decoupling constraints.

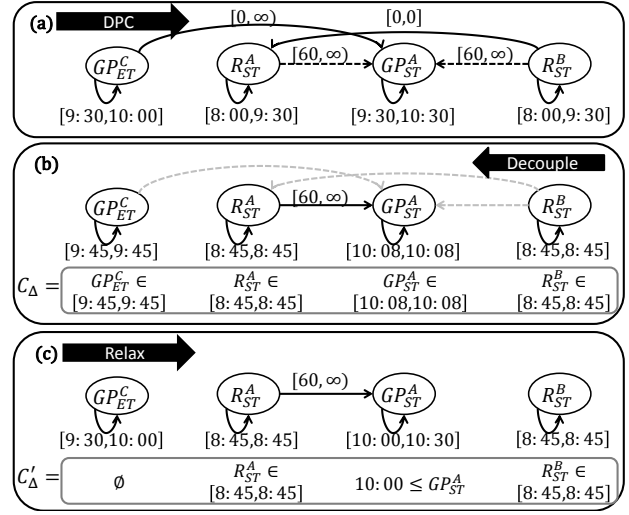


Figure 2: Applying the MaTDP algorithm to the example scheduling problem.

To avoid inconsistency due to concurrency, before calculating decoupling constraints for v_k , an agent blocks in line 8 until it receives the fresh, newly computed weights w_{zj}, w_{jz} from v_j 's agent ($\text{Agent}(v_j)$, as sent in line 14) for each external edge $e_{jk} \in E_X^i$ where $j > k$. While this implies some sequentialization, it also allows for concurrency whenever variables do not share an external edge. For example, in Figure 2(b), because GP_{ET}^C and R_{ST}^A do not share an edge, after Ann’s agent has assigned GP_{ST}^A , both Ann and Chris’ agents can concurrently and independently update and assign R_{ST}^A and GP_{ET}^C respectively. Finally, each agent establishes PPC in response to its new decoupling constraints, by executing P3C-2 (which refers to the second phase of the single-agent P3C algorithm presented in [6] as Algorithm 3).

THEOREM 1. *The MaTDP algorithm has an overall time complexity of $O(|V|\omega_o^{*2})$ and requires $O(|E_X|)$ messages.*

PROOF. The MaTDP algorithm calculates DPC and PPC in $O(|V|\omega_o^{*2})$ time. Unary, decoupling constraints are calculated for each of $|V_X|$ external variables $v_k \in V_X$ (lines 4-16), after iterating over each of v_k 's $O(\omega_o^*)$ neighbors (lines 6-12). Thus decoupling requires $O(|V|\omega_o^*) \subseteq O(|V|\omega_o^{*2})$ time, and so MaTDP has an overall time complexity of $O(|V|\omega_o^{*2})$. The MaTDP algorithm sends exactly one message for each external constraint in line 14, for a total of $O(|E_X|)$ messages. \square

THEOREM 2. *The MaTDP algorithm is sound.*

PROOF. Lines 1-2 return INCONSISTENT whenever the input MaSTP \mathcal{M} is not consistent. By contradiction, assume that there exists some external constraint c_{xy} with bound b_{xy} that is *not* satisfied when the decoupling constraints c_{xz} and c_{zy} , calculated by MaTDP with bounds b_{xz} and b_{zy} respectively, are (that is $b_{xz} + b_{zy} > b_{xy}$). WLOG, let $x < y$ in o_X . Notice, line 1 (DPC) implies $w_{xy} \leq b_{xy}$. Line 11 then implies $w_{xz} + w_{zy} \leq w_{xy} \leq b_{xy}$ (since inductively $w_{yz} + w_{zy} = 0$). Notice that after line 11, all other possible updates to w_{xz} that occur before c_{xz} is constructed in line 15 (e.g., in lines 10-11, 13) only tighten (never relax) w_{xz} , and

so $b_{xz} + b_{zy} \leq w_{xz} + w_{zy} \leq w_{xy} \leq b_{xy}$. However, this is a contradiction to our assumption that $b_{xz} + b_{zy} > b_{xy}$, so the decomposable distance graph and constraints C_Δ calculated by MaTDP form a temporal decoupling of \mathcal{M} . \square

THEOREM 3. *The MaTDP algorithm is complete.*

PROOF (SKETCH). The basic intuition for this proof is provided by the fact that, in some sense, the MaTDP algorithm is simply a distributed version of the basic backtrack-free assignment procedure that can be applied to a DPC distance graph. We show that when we choose bounds for new, unary decoupling constraints for v_k (effectively in line 13), w_{zk}, w_{kz} are path consistent with respect to all other variables. This is because not only is the distance graph DPC, but also the updates in lines 10-11 guarantee that w_{zk}, w_{kz} are path consistent with respect to v_k for all $j > k$ (since each such path from v_j to v_k will be represented as an edge e_{jk} in the distance graph). So the only proactive edge tightening that occurs, which happens in line 13 and guarantees that $w_{zk} + w_{kz} = 0$, is done on path consistent edges and thus will never introduce a negative cycle (or empty domain). \square

3.2 A Minimal Temporal Decoupling Relaxation Algorithm

The goal of the Multiagent Temporal Decoupling Relaxation (MaTDR) algorithm, presented as Algorithm 2, is to replace the set of decoupling constraints produced by the MaTDP algorithm, C_Δ , with a set of *minimal* decoupling constraints, C'_Δ . Recall that a minimal decoupling is one where, if the bound of any decoupling constraint $c \in C'_\Delta$ for some agent i is relaxed, then $\{S_{L+\Delta}^1, S_{L+\Delta}^2, \dots, S_{L+\Delta}^n\}$ is no longer a decoupling. Clearly the temporal decoupling produced when running MaTDP using the default heuristic on our example problem, as shown in Figure 2(b), is not minimal. The basic idea of the MaTDR algorithm is to revisit each external timepoint v_k and, while holding the domains of all other external timepoint variables constant, relax the bounds of v_k 's decoupling constraints as much as possible.

The MaTDR works in original o_X order, and thus starts with GP_{ET}^C . First, Chris' agent removes GP_{ET}^C 's decoupling constraints and restores GP_{ET}^C 's domain to [9:30,10:00] by updating the corresponding edge weights to their stored, DPC values (lines 1,3). Notice that lines 3-16 are similar to backwards execution of lines 6-12 in the MaTDP algorithm, except that a separate, "shadow" δ bound representation is used and updated only with respect to the original external *constraint bounds* (not edge weights). Also, in lines 17-24, a decoupling constraint is *only* constructed when the bound of the potential new constraint (e.g. δ_{kz}) is tighter than the already implied edge weight (e.g. when $\delta_{kz} < w_{kz}$). So in the case of GP_{ET}^C , the only constraint involving GP_{ET}^C is that it should occur before GP_{ST}^A . However, GP_{ST}^A is currently set to occur at 10:08 ($\delta=10:08$), and since GP_{ET}^C is already constrained to occur before 10:00 ($w=10:00$), $\delta \not\prec w$, and so no decoupling constraints are added to the set C'_Δ for GP_{ET}^C . The next variable to consider is R_{ST}^A , whose domain relaxes back to [8:00,9:30]. However, since R_{ST}^A shares a synchronization constraint with R_{ST}^B , whose current domain is [8:45,8:45], Ann's agent will end up re-enforcing the original decoupling constraints of $R_{ST}^A \in [8:45,8:45]$. On the other hand, after Ann's agent recovers GP_{ST}^A 's original DPC domain of [9:30,10:30], it then needs to ensure that GP_{ST}^A will always occur after GP_{ET}^C 's new domain of [9:30,10:00]. In

Algorithm 2 Multiagent Temporal Decoupling Relaxation (MaTDR)

Input: \mathcal{G}^i , and the DPC weights, $w_{zk}^{DPC}, w_{kz}^{DPC}$, for each $v_k \in V_X^i$
Output: C'_Δ , agent i 's *minimal* decoupling constraints, and \mathcal{G}^i , agent i 's PPC distance graph w.r.t. C'_Δ .

- 1: $C'_\Delta \leftarrow \emptyset$
- 2: **for** $k = 1 \dots n$ such that $v_k \in V_L^i$ **do**
- 3: $w_{zk} \leftarrow w_{zk}^{DPC}, w_{kz} \leftarrow w_{kz}^{DPC}$
- 4: $\delta_{zk} \leftarrow \delta_{kz} \leftarrow \infty$
- 5: **for** $j = 1$ to n such that $\exists e_{jk} \in E_L^i \cup E_X$ **do**
- 6: **if** $e_{jk} \in E_X^i$ **then**
- 7: **if** $j < k$ **then** $w_{zj}, w_{jz} \leftarrow$ Block receive from *Agent*(v_j)
- 8: **if** c_{jk} exists **then** $\delta_{zk} \leftarrow \min(\delta_{zk}, b_{jk} - w_{jz})$
- 9: **if** c_{kj} exists **then** $\delta_{kz} \leftarrow \min(\delta_{kz}, b_{kj} - w_{zj})$
- 10: **else if** $j < k$ **then**
- 11: $w_{zk} \leftarrow \min(w_{zk}, w_{zj} + w_{jk})$
- 12: $w_{kz} \leftarrow \min(w_{kz}, w_{kj} + w_{jz})$
- 13: **end if**
- 14: **end for**
- 15: **if** $\delta_{kz} < w_{kz}$ **then**
- 16: $w_{kz} \leftarrow \delta_{kz}$
- 17: $C'_\Delta \leftarrow C'_\Delta \cup \{(z - v_k \leq \delta_{kz})\}$
- 18: **end if**
- 19: **if** $\delta_{zk} < w_{zk}$ **then**
- 20: $w_{zk} \leftarrow \delta_{zk}$
- 21: $C'_\Delta \leftarrow C'_\Delta \cup \{(v_k - z \leq \delta_{zk})\}$
- 22: **end if**
- 23: Send w_{zk}, w_{kz} to each *Agent*(v_j) s.t. $j > k, e_{jk} \in E_X^i$
- 24: **end for**
- 25: **return** \mathcal{G}^i, C'_Δ

this case, decoupling from GP_{ET}^C requires only a lower bound of 10:00 for GP_{ST}^A and results in a more flexible domain of [10:00,10:30]. The minimal decoupling constraints and corresponding distance graph that MaTDR calculates for the running example are presented in Figure 2(c) and Figure 1(c).

The type of update performed on timepoint v_k 's actual domain edge weights, w_{zk} and w_{kz} (line 11-12), and shadow edge weights, δ_{zk} and δ_{kz} (line 8-9), differs based on whether the edge e_{jk} being considered in the inner loop is local or external respectively. For example, suppose v_k has a domain of [1:00,4:00], v_j has a domain of [2:00,2:30] (which already incorporates its new decoupling constraints, since v_j appears before v_k in o_X), and e_{jk} has the label [0,60] (e.g., $v_k - v_j \in [0,60]$), which corresponds to bounds of original constraints. If e_{jk} is an external edge, the "shadow" domain of v_k would be updated by lines 8-9 to be [2:30,3:00]. Otherwise, if e_{jk} is a local edge, then the actual domain of v_k would be instead updated by lines 11-12 and result in the less restrictive domain [2:00, 3:30]. The difference between the two updates is that the updates in lines 8-9 guarantee that *all* possible assignments to the two variables will be consistent with respect to the external constraint, whereas the updates in lines 11-12 only guarantee that there exists *some* local assignment to the two variables that will be consistent. Finally, notice that if the domain of v_j had instead been assigned (e.g., to [2:30,2:30]), the updates in lines 8-9 and lines 11-12 would have resulted in the exact same update to the domain of v_k (e.g., [2:30,3:30]). Due to its similarity to the MaTDP algorithm, we forgo formally proving the correctness and computational complexity of the MaTDR sub-routine.

THEOREM 4. *The reference constraints calculated by the MaTDR algorithm form a minimal temporal decoupling of S .*

PROOF (SKETCH). The proof that C'_Δ form a temporal

decoupling is roughly analogous to the proof for Theorem 3.1. By contradiction, we show that if the bound b_{xz} of some decoupling constraint $c_{xz} \in C'_\Delta$ is relaxed by some small, positive value $\epsilon_{xz} > 0$, then C'_Δ is no longer a temporal decoupling. This is because lines 8-9 imply that there exists some y such that either, $b_{xz} = b_{xy} - b_{zy}$, and thus $b_{xz} + \epsilon_{xz} + b_{zy} > b_{xy}$ (and thus no longer a temporal decoupling), or that $b_{zy} = b_{xy} - (b_{xz} + \epsilon_{xz})$ (and so is either not a decoupling or requires us to also alter b_{zy} in order to maintain the temporal decoupling). \square

3.3 Privacy

The natural distribution of the MaSTP representation affords a partitioning of the MaSTP into private and shared components [2]. Agent i 's set of *private variables*, V_P^i , is the subset of agent i 's local variables that are involved in *no* external constraints, $V_P^i = V_L^i \setminus V_X$. Agent i 's set of *private constraints*, C_P^i , is the subset of agent i 's local constraints C_L^i that include at least one of its private variables. Alternatively, the *shared STP*, $\mathcal{S}_S = \langle V_S, C_S \rangle$ is composed of the set of *shared variables*, V_S , where $V_S = V_X \cup \{z\}$, and the set of *shared constraints*, C_S , where shared constraints are defined exclusively over shared variables and by definition include all external constraints, $C_S = C_X \cup \{\bigcup_i C_N^i \setminus C_P^i\}$. In the example displayed in Figure 1, all shared variables and constraints are represented with dashed lines. \mathcal{S}_S represents the maximum portion of the MaSTP that a set of colluding agents could infer, given only the joint MaSTP specification [2]. Hence, given the distribution of an MaSTP \mathcal{M} , if agent i executes a multiagent algorithm that does not reveal any of its private timepoints or constraints, it can be guaranteed that any agent $j \neq i$ will not be able to infer any private timepoint in V_P^i or private constraint in C_P^i by also executing the multiagent algorithm — at least not without requiring conjecture or ulterior (methods of inferring) information on the part of agent j . Additionally, it is not generally required that any agent knows or infers the entire shared STP. In our algorithms, agents attempt to minimize shared knowledge to increase efficiency.

COROLLARY 5. *The MaTDP and MaTDR algorithms never reveal any of agent i 's private variables or private constraints (or edges) and hence maintain privacy over them.*

PROOF (SKETCH). Follows from proof of the properties of the MaSTP privacy partitioning, Theorem 1 in [2]. \square

Together, the MaTDP and MaTDR algorithms calculate a minimal temporal decoupling for an MaSTP. In the next section, we empirically compare the performance of these algorithms with previous approaches.

4. EMPIRICAL EVALUATION

In the following subsections, we introduce the methodology we use to empirically evaluate the performance of our algorithm's computational effort and flexibility.

4.1 Methodology

To develop results comparable to those elsewhere in the literature, we model our experimental setup after [2] and [4], adapting the random problem generator described in [4] so that it generates MaSTP instances. Each problem instance has A agents each with start timepoints and end timepoints

for 10 actions. Each action is constrained to occur within the time interval $[0,600]$ relative to a global zero reference timepoint, z . Each activity's duration is constrained by a lower bound, lb , chosen uniformly from interval $[0,60]$ and an upper bound chosen uniformly from the interval $[lb, lb + 60]$. In addition to these constraints, the generator adds 50 additional local constraints for each agent and N total external constraints. Each of these additional constraints, e_{ij} , is constrained by a bound chosen uniformly from the interval $[-B_{ji}, B_{ij}]$, where v_i and v_j are chosen, with replacement, with uniform probability. To confirm the significance of our results, we generate and evaluate the expected performance of our algorithms over 25 independently generated trials for each parameter setting. Since the novelty of our algorithms lies within the temporal decoupling aspects of the problem, we only generate consistent MaSTP problem instances to compare the computational effort of full applications of the various decoupling algorithms. We modeled a concurrently executing multiagent system by systematically sharing a 3 Ghz processor with 4 GB of RAM by interrupting each agent after it performed a single bound operation (either an update or evaluation) and a single communication (sending or receiving one message).

4.2 Evaluation of Computational Effort

In the first set of experiments, we empirically compared:

- **MaTDP+R** – our MaTDP algorithm with the MaTDR subroutine,
- **Cent. MaTDP+R** – a single agent that executes MaTDP+R on a centralized version of the problem,
- **D-P3C** – our implementation of the D Δ P3C distributed algorithm for establishing PPC for an MaSTP (but not a decoupling) [2], and
- **TDP** – our implementation of the fastest variation (the RGB variation) of the (centralized) TDP algorithm as reported in [4].

For the TDP approach, we used the Floyd-Warshall algorithm to initially establish FPC and the incremental update described in [5] to maintain FPC as new constraint were posted. We evaluated approaches across two metrics. The non-concurrent computation (**NCC**) metric is the number computational cycles before all agents in our simulated multiagent environment have completed their execution of the algorithm [2]. The other metric we report in this section is the total number of messages exchanged by agents.

In the first experiment set (Figure 3), $A = \{1, 2, 4, 8, 16, 32\}$ and $N = 50 \cdot (A - 1)$. In the second experiment set (Figure 4), $A = 25$ and $N = \{0, 50, 100, 200, 400, 800, 1600, 3200\}$. The results shown in both figures demonstrate that our MaTDP+R algorithm clearly dominates the original TDP approach in terms of execution time, even when the MaTDP+R algorithm is executed in a centralized fashion. When compared to the centralized version of the MaTDP+R algorithm, the distributed version has a speedup (centralized computation/distributed computation) that varies between 19.4 and 24.7. This demonstrates that the structures of the generated problem instances support parallelism and that the distributed algorithm can exploit this structure to achieve significant amounts of parallelism.

Additionally, notice that the MaTDP+R algorithm dominates the D Δ P3C algorithm in both computation and number

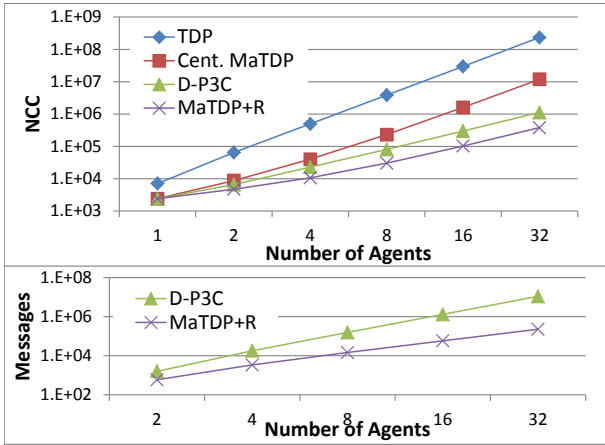


Figure 3: Computational effort as A grows.

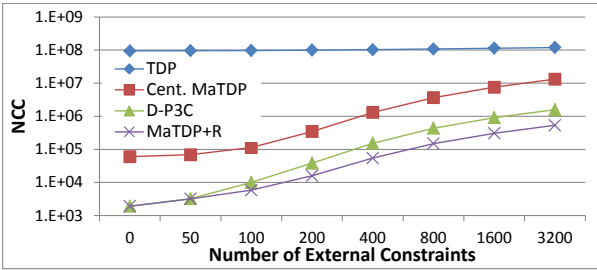


Figure 4: Computational effort as N increases.

of messages (which held true in both experiments, although messages are not displayed in Figure 4 due to space considerations), which means the MaTDP+R algorithm can calculate a temporal decoupling with less computational effort than the D Δ P3C algorithm can calculate a decomposable, PPC representation of the MaSTP. This is due to the fact that, while the MaTDP+R is generally bound by the same runtime complexity as the D Δ P3C, as argued in Theorem 1, the complexity of the actual decoupling procedure is less in practice, since the decoupling algorithm only calculates new bounds for reference edges, instead of calculating new bounds for *every* shared edge. This is important because if agents instead chose to try to maintain the complete set of consistent joint schedules (as represented by the decomposable, PPC output of D Δ P3C), agents may likely perform *additional* computation and communication *every* time a new constraint arises, whereas the agents that calculate a temporal decoupling can perform all additional computation locally and independently, unless or until a new constraint arises that invalidates the temporal decoupling. The fact that MaTDP+R algorithm dominates the D Δ P3C algorithm also implies that even if the original TDP algorithm were adapted to exploit the current state-of-the-art distributed PPC algorithm [2], our algorithm would still dominate the basic approach in terms of computational effort. Overall, we confirmed that we could exploit the structure of the MaSTP to calculate a temporal decoupling not only more efficiently than previous TDP approaches, but also in a distributed manner, avoiding centralization costs previously required, and exploiting parallelism to lead to impressive levels of speedup.

We next ask whether the quality of our MaTDP+R algorithm is competitive.

4.3 Evaluation of Flexibility

As mentioned earlier, one of the key properties of a decomposable MaSTP is that it can represent a set of consistent joint schedules, which in turn can be used as a hedge against scheduling uncertainty. In the following subsections we describe a metric for more generally quantifying the robustness of an MaSTP, and hence a temporal decoupling, in terms of a *flexibility* metric and perform an empirical evaluation of our algorithms with regards to flexibility.

4.3.1 Flexibility Metrics

Hunsberger introduced two metrics, *flexibility* ($Flex$) and conversely *rigidity* (Rig), that quantify the basic notion of robustness so that the quality of alternative temporal decouplings can be compared [4]. He defined the flexibility between a pair of timepoints, v_i and v_j , as the sum $Flex(v_i, v_j) = B_{ij} + B_{ji}$ which is always positive for consistent MaSTPs. The rigidity of a pair of timepoints is defined as $Rig(v_i, v_j) = \frac{1}{1 + Flex(v_i, v_j)}$, and the rigidity over an entire STP is the root mean square (RMS) value over the rigidity value of *all* pairs of timepoints:

$$Rig(S) = \sqrt{\frac{2}{|V|(|V| + 1)} \sum_{i < j} [Rig(v_i, v_j)]^2}.$$

This implies that $Rig(S) \in [0, 1]$, where $Rig(S) = 0$ when S has no constraints and $Rig(S) = 1$ when S has a single solution [4]. Since $Rig(S)$ requires FPC to calculate, we only apply this metric as a post-processing evaluation technique by centralizing and establishing FPC on the temporal decouplings returned by our algorithms. There exists a centralized, polynomial time algorithm for calculating an optimal temporal decoupling [7], but it requires an evaluation metric that is a linear function of distance graph edge weights, which the aggregate rigidity function $R(S)$, unfortunately, is not.

4.3.2 Evaluation

In our second set of experiments, we compare the rigidity of the temporal decouplings calculated by:

- **Default** – a variant MaTDP algorithm that uses the the described, default heuristic, but without MaTDR,
- **Relaxation** – the default MaTDP with MaTDR,
- **Locality** – a variant of the MaTDP algorithm where, in line 13, agents heuristically bias how much they tighten w_{zk} relative to w_{kz} using information from applying the *full* D Δ P3C algorithm in line 1 (no MaTDR),
- **All** – the MaTDP using both the locality heuristic and the MaTDR sub-routine,
- **Input** – the rigidity of the input MaSTP, and
- **TDP** – our implementation of Hunsberger’s RLF variation of his TDP algorithm (where $r = 0.5$ and $\epsilon = 1.0$ which lead to a computational multiplier of approximately 9) that was reported to calculate the least rigid decoupling in [4].

In this experiment, $A = 25$ and $N = \{50, 200, 800\}$. Table 1 displays the rigidity of the temporal decoupling calculated by each approach. On average, as compared to the Default, the Relaxation approach decreases rigidity by 51.0% (while

Table 1: The rigidity values of various approaches.

	N=50	N=200	N=800
Input	0.418	0.549	0.729
All	0.508	0.699	0.878
Relaxation	0.496	0.699	0.886
Locality	0.621	0.842	0.988
Default	0.628	0.849	0.988
TDP	0.482	0.668	0.865

increasing computational effort by 30.2%), and the Locality approach decreases rigidity by 2.0% (while increasing computational effort by 146%). The Relaxation approach, which improves the output decoupling the most, offers the best return on investment. The locality heuristic, however, is very computationally expensive while providing no significant improvement in rigidity. We also explored combining these rigidity decreasing techniques, and while the increase in computational effort tended to be additive (the All approach increases effort by 172%), the decrease in rigidity did not. In fact, no heuristics or other combinations of techniques led to a statistically significant decrease in rigidity (as compared to the default, Relaxation approach) in the cases we investigated. The All approach decreased rigidity by only 49.9% in expectation.

The fact that the Relaxation approach alone decreases rigidity by more than any other combination of other approaches can be attributed to both the structure of an MaSTP and how rigidity is measured. First, the Relaxation improves the distribution of flexibility to the shared timepoints reactively, instead of proactively trying to guess good values. As the MaTDP algorithm tightens bounds, the general triangulated graph structure formed by the elimination order “branches out” the impact of this tightening. So if the first timepoint is assigned, this defers more flexibility to the subsequent timepoints that depend on the bounds of the first timepoint, of which there could be many. So by being proactive, other heuristics may steal flexibility from a greater number of timepoints, whereas the MaTDR algorithm allows this flexibility to be recovered only after the (possibly many more) subsequent timepoints have set their bounds to maximize their local flexibility.

Notice from Table 1 that the TDP approach decreases the rigidity the most, representing on average a 20.6% decrease in rigidity as compared to the Relaxation approach. However, this additional reduction in rigidity comes at a significant computational cost — the TDP approach incurs, in expectation, over 10,000 *times* more computational effort than our Relaxation approach. While in some scheduling environments the costs of centralization (e.g. privacy) alone would invalidate this approach, in others the computational effort may be prohibitive if constraints arise faster than the centralized TDP algorithm can calculate a temporal decoupling. Further, in many scheduling problems, all temporal decouplings may be inherently rigid if, for example, many of the external constraints enforce synchronization (e.g. Ann’s run start time), which requires fully assigning timepoints in order to decouple. Overall, the Relaxation approach, in expectation, outputs a *high-quality* temporal decoupling, approaching the quality (within 20.6%) of the state-of-the-art centralized approach [4], in a *distributed, privacy-maintaining* manner *faster* than the state-of-the-art MaSTP solution algorithms.

5. CONCLUSION

In this paper, we have presented a new, distributed algorithm that solves the MaTDP without incurring the costs of centralization like previous approaches. We have proven that the MaTDP algorithm is correct, and demonstrated both analytically and empirically that it calculates a temporal decoupling faster than previous approaches, exploiting sparse structure and parallelism when it exists. Additionally we have introduced the MaTDR algorithm for relaxing the bounds of existing decoupling constraints to form a *minimal* temporal decoupling, and empirically showed that this algorithm can decrease rigidity by upwards of 50% (within 20.6% of the state-of-the-art centralized approach) while increasing computational effort by as little as 20%. Overall, we have shown that the combination of the MaTDP and MaTDR algorithms calculates a temporal decoupling faster than state-of-the-art distributed MaSTP solution algorithms and the MaTDR algorithm reduces rigidity further than other heuristics we evaluated. In the future, we hope to evaluate the computational and communication costs of our algorithms in the context of a dynamic scheduling environment. We hope to extend the MaTDR algorithm to an anytime approach for recovering flexibility as new constraints arise and evaluate the computational effort in comparison with calculating a new temporal decoupling after an existing temporal decoupling becomes inconsistent. Additionally, we hope to develop additional flexibility metrics that can be evaluated in a distributed setting for heuristically guiding scheduling agents in dynamic scheduling environments.

6. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their comments and suggestions. This work was supported, in part, by the NSF under grants IIS-0534280 and IIS-0964512 and by the AFOSR under Contract No. FA9550-07-1-0262.

7. REFERENCES

- [1] C. Bliet and D. Sam-Haroud. Path Consistency on Triangulated Constraint Graphs. In *Proc. of IJCAI-99*, pages 456–461, 1999.
- [2] J. Boerkoel and E. Durfee. A Comparison of Algorithms for Solving the Multiagent Simple Temporal Problem. In *Proc. of ICAPS-10*, pages 26–33, 2010.
- [3] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. In *Knowledge representation*, volume 49, pages 61–95. The MIT Press, 1991.
- [4] L. Hunsberger. Algorithms for a temporal decoupling problem in multi-agent planning. In *Proc of AAAI-02*, pages 468–475, 2002.
- [5] L. Planken. Incrementally solving the stp by enforcing partial path consistency. In *Proc. of PlansSIG-08*, pages 87–94, 2008.
- [6] L. Planken, M. de Weerd, and R. van der Krogt. P3C: A new algorithm for the simple temporal problem. In *Proc. of ICAPS-08*, pages 256–263, 2008.
- [7] L. Planken, M. de Weerd, and C. Witteveen. Optimal temporal decoupling in multiagent systems. In *Proc. of AAMAS-10*, pages 789–796, 2010.
- [8] L. Xu and B. Choueiry. A new efficient algorithm for solving the simple temporal problem. In *Proc. of TIME-ICTL-03*, pages 210–220, 2003.

Decomposing constraint systems: Equivalences and computational properties

Wiebe van der Hoek
Dept. of Computer Science
University of Liverpool,
United Kingdom
wiebe@csc.liv.ac.uk

Cees Witteveen
Dept. of Software Technology
Delft University of Technology,
The Netherlands
C.Witteveen@tudelft.nl

Michael Wooldridge
Dept. of Computer Science
University of Liverpool,
United Kingdom
mjw@liverpool.ac.uk

ABSTRACT

Distributed systems can often be modeled as a collection of distributed (system) variables whose values are constrained by a set of constraints. In distributed multi-agent systems, the set of variables occurring at a site (subsystem) is usually viewed as controllable by a local agent. This agent assigns values to the variables, and the aim is to provide distributed methods enabling a set of agents to come up with a global assignment (solution) that satisfies all the constraints. Alternatively, the system might be understood as a distributed database. Here, the focus is on ensuring consistency of the global system if local constraints (the distributed parts of the database) change. In this setting, the aim is to determine whether the existence of a global solution can be guaranteed. In other settings (e.g., P2P systems, sensor networks), the values of the variables might be completely out of control of the individual systems, and the constraints only characterize globally normal states or behavior of the system. In order to detect anomalies, one specifies distributed methods that can efficiently indicate violations of such constraints. The aim of this paper is to show that the following three main problems identified in these research areas are in fact identical: (i) the problem of ensuring that independent agents come up with a global solution; (ii) the problem of ensuring that global consistency is maintained if local constraint stores change; and (iii) the problem of ensuring that global violations can be detected by local nodes. This claim is made precise by developing a decomposition framework for distributed constraint systems and then extracting *preservation properties* that must be satisfied in order to solve the above mentioned problems. Although satisfying the preservation properties seems to require different decomposition modes, our results demonstrate that in fact these decomposition properties are equivalent, thereby showing that the three main problems identified above are identical. We then show that the complexity of finding such decompositions is polynomially related to finding solutions for the original constraint system, which explains the popularity of decomposition applied to tractable constraint systems. Finally, we address the problem of finding optimal decompositions and show that even for tractable constraint systems, this problem is hard.

Cite as: Decomposing constraint systems, Wiebe van der Hoek, Cees Witteveen and Michael Wooldridge, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 149-156.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent Systems; I.2.4 [Knowledge representation formalisms and methods]

General Terms

Theory

Keywords

distributed constraint systems, decomposition, complexity

1. INTRODUCTION

Distributed systems can frequently be modelled as a collection of distributed (system) variables X , whose values are constrained by a set of constraints. Usually, one distinguishes a set of sites, each of which contains a disjoint subset of variables $X_i \subseteq X$. Each site is responsible for those constraints relating to the variables that occur in its variable set X_i (its set of *local constraints*). Sites must interact with respect to the set of *global constraints*¹, which relates variables of different components X_i . The way such a model of a distributed system is used largely depends on the application domain. We can identify at least three different application domains: multi-agent systems, distributed databases, and P2P systems.

In multi-agent systems, one typically assumes that there is a set of agents, with each agent A_i controlling the variables in X_i . It is the common task of all the agents to assign suitable values to their variables such that all constraints are satisfied. Since each agent tries to assign values to the variables independently from the others, only the constraints whose variables occur in the agent's control set are guaranteed to be satisfied. The main research problem is then to provide (distributive) methods enabling the agents to come up with a global assignment (solution) satisfying all constraints – both local and global.

In other domains, such as databases, such a distributed constraint system is conceived as a model of a distributed database. Here, the focus is not on finding solutions for a fixed set of constraints, but to ensure consistency of the

¹Although the term “global constraints” in the CP-literature refers to constraints encapsulating sets of other constraints, in the context of distributed constraint processing we use this term only to distinguish them from local constraints. That is, global constraints are those constraints whose variables occur in more than one control set.

global system in the event that local constraints (the distributed components of the database) change. Such a notion of consistency is expressed by global integrity constraints that need to be respected whatever changes take place at local sites. Hence, here the issue is not giving values to variables such that all constraints are satisfied, but to ensure that there exists at least one possibility for such a globally satisfying assignment.

While in the above mentioned areas the focus is on finding a solution or guaranteeing the existence of at least one solution, in areas such as peer-to-peer (P2P) systems and sensor networks, modeling by distributed constraint systems is often focused on the detection of constraint *violations*. Here, the set of constraints is used to characterize acceptable states or behavior of the global system, and violations of such constraints indicate potentially problematic anomalies (e.g., a DDoS attack). In order to detect such anomalies, one specifies a set of distributed methods that aim to establish violations of such constraints as efficiently as possible.

Although these problems arise in different areas they all have a common aspect. From an abstract point of view, one might consider the set of agents (sites, local databases) with their constraints as a *decomposition* of the original constraint system induced by a partitioning of the variables. In all three areas mentioned above, the problem is how we might ensure this decomposition to have some *preservation properties* with respect to the underlying global system.

From a multi-agent systems point of view, in distributed constraint systems one often considers the local agent as autonomous. Here, decomposition has to ensure that the local constraints can be solved completely independently from the others, after which the local solutions can always be *merged* to yield a solution to the complete system. Hence, in multi-agent systems research the focus of decomposition has been on a *solution preserving* property: in obtaining a global solution, local solutions should always be preserved in order to ensure independent local problem solving. For example, in [10] decomposition² has been applied to ensure that independently chosen schedules for subnetworks of a Simple Temporal Network (STN) can always be merged to a joint schedule of the total network. In [11] a decomposition technique is presented to ensure decentralized cooperative control of multi-agent systems where satisfaction of all (distributed) subtasks of a joint task implies the fulfillment of the complete task as well.

In the database community, one wants to ensure that whenever each local database is consistent, the consistency of the global database is implied, whatever changes occur locally. The method applied here is to provide *localized versions* of global integrity constraints that ensure that, whatever local information satisfying these constraints is added to the (distributed) database, the global consistency of the total database will be preserved [2, 9, 4]. Hence the database community is interested in *consistency preserving* decompositions³.

²In this paper, Hunsberger has adopted the term *temporal decoupling* for decomposition in STNs.

³Quite closely related to the database community, in the sensor network community, one distinguishes the *localization* problem, where a distributed constraint is reformulated into local constraints for mobile entities and is adjusted dynamically [12, 15]. The satisfaction of the distributed constraint is guaranteed whenever all the local constraints are satisfied.

The P2P community aims at the efficient detection of constraint *violations*. Here, normal operations are specified by a global constraint C_S . For reasons of efficiency, one prefers not to monitor all sites to establish violations of these constraints. Therefore, localizations of such constraints are provided to each node, such that it has its own violation detection mechanism [1]. A global violation detection mechanism is triggered only if some local node detects a violation, thus saving communication between the nodes. So, in the P2P community, one is interested in *safety preserving* decompositions of integrity constraints, ensuring that whenever all local states indicate safeness (no violation detection occurs) of their local states, the global state is safe (that is, the global integrity constraints are all respected), too.

In summary, it seems that we can study all three problems in a common decomposition framework, where the only difference between these problems is in their preservation properties. In fact, as, we pointed out, there has been extensive research in these three areas, focusing on either the solution preserving, the consistency preserving, or the safety preserving aspect of decompositions in distributed constraint systems. However, to the best of our knowledge, there have been no attempts to establish their equivalence. In short, the aim of the present paper is to address this issue and to investigate some computational aspects of such decompositions.

We begin in Section 2 by presenting a formal framework for investigating these properties, and present the technical preliminaries used in the remainder of the paper. In Section 3, we investigate the relationships between the properties we identified, i.e., we consider whether the properties are independent from each other, whether they imply each other, or whether they are they completely identical. In Section 4, we address some computational aspects of these preservation properties — for example, how difficult is it to find a {solution, consistency, violation}-preserving decomposition. We will establish some tight computational connections between these problems and the general problem of finding a solution to a constraint system. Then, in Section 5, we will address the problem of *information loss* inherent in solving a decomposed problem as opposed to solving the problem at a global level. We will indicate that in general the problem of establishing the exact information loss is intractable. Finally, in Section 6, we state some final conclusions to place this work into a broader perspective.

2. PRELIMINARIES

In this section we briefly define constraint systems, distributed constraint systems, and decompositions of distributed constraint systems.

2.1 (Distributed) Constraint Systems

A constraint system is a tuple $\mathcal{S} = \langle X, D, C \rangle$ where X is a (finite) set of *variables*, D is a set of (value) *domains* D_i for every variable $x_i \in X$, and C is a set of *constraints* over X . We assume constraints $c \in C$ to be specified as formulas of some language. We will not require any specific language for constraints $c \in C$, but it is useful to assume some fundamental properties. Specifically, we will assume the language contains constants for elements in the domains D_i , the usual Boolean connectives (\neg, \vee, \wedge), and equality. We also require that it is possible to determine whether a solution satisfies a constraint in polynomial time. Formally,

a solution σ of the system $\mathcal{S} = \langle X, D, C \rangle$ is an assignment $\sigma = \{x_i \leftarrow d_i\}_{i=1}^n$ to all variables in X such that $d_i \in D_i$, and each constraint $c \in C$ is satisfied. We sometimes write $\sigma \models C$ to mean that σ is a solution to C . Given a subset $X_i \subseteq X$, we let σ_{X_i} denote the restriction of σ to the subset X_i . Where no confusion is possible, we will use σ_i as a shorthand for σ_{X_i} . The set of all assignments σ is denoted by Σ . Likewise, the set of all assignments for a subset $X_i \subseteq X$ is denoted by Σ_{X_i} or Σ_i .

The set of solutions σ to system $\mathcal{S} = \langle X, D, C \rangle$ will be denoted by $Sol(\mathcal{S})$. \mathcal{S} is called *consistent* if $Sol(\mathcal{S}) \neq \emptyset$. For every $c \in C$, let $Var(c)$ denote the set of variables mentioned in c . For a set of constraints C , we let

$$Var(C) = \bigcup_{c \in C} Var(c).$$

Given $\mathcal{S} = \langle X, D, C \rangle$, we obviously require $Var(C) \subseteq X$. If D is a set of value domains D_i for variables $x_i \in X$ and $X' \subset X$ then $D_{X'}$ is the set of value domains D_i of the variables $x_i \in X'$. Likewise, given a set of constraints C and a set of variables X' , we let $C_{X'}$ denote the subset $\{c \in C \mid Var(c) \subseteq X'\}$ of constraints over X' .

In this paper we consider constraint systems \mathcal{S} that are *distributed* [19]; that is, there is a set of N agents A_i , each being able to make assignments to, or to add constraints over a subset X_i of the set X of variables. Here, we assume that agents do not share control over the variables, and that every variable is controlled by an agent. Hence, the collection $\{X_i\}_{i=1}^N$ constitutes a *partitioning* of X , i.e.:

- $\bigcup_{i=1}^N X_i = X$; and
- for all $1 \leq i < j \leq N$, $X_i \cap X_j = \emptyset$.

To indicate that a constraint system $\mathcal{S} = \langle X, D, C \rangle$ is distributed by a partitioning $\{X_i\}_{i=1}^N$ of X , we write $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ and call it a *distributed constraint system*. We are particularly interested in those distributed systems $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ where each agent A_i , controlling the set X_i , only processes a set of constraints over X_i , and does not take into account other constraints. That effectively implies that in such a case, instead of one constraint system \mathcal{S} and a partitioning $\{X_i\}_{i=1}^N$, we have a *set* of independent constraint systems $\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle$, where each C'_i is a set of constraints over X_i , i.e., $Var(C'_i) \subseteq X_i$. We call the resulting set $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ of such subsystems a *decomposed constraint system*⁴. We say that σ is a solution of \mathcal{S}' if, for each i , σ_i is a solution of $\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle$. The decomposition \mathcal{S}' is said to be consistent if $\bigcup_i C'_i$ is consistent.

Whenever $\{X_i\}_{i=1}^N$ is a partitioning of X , we will write $\sigma = \sigma_1 \sqcup \sigma_2 \sqcup \dots \sqcup \sigma_N$ to indicate an assignment σ that is composed of the disjoint assignments σ_i for X_i . Likewise, we will write $Sol(\mathcal{S}_1) \sqcup Sol(\mathcal{S}_2) \sqcup \dots \sqcup Sol(\mathcal{S}_N)$ to indicate the set of global assignments σ that can be constructed by simply composing all local solutions of the subsystems \mathcal{S}_i .

2.2 Decompositions: preservation properties

We now discuss three specifications of the relationship between a distributed constraint system \mathcal{S} and a decomposition \mathcal{S}' , with respect to the three preservation properties we informally discussed above.

⁴For the moment, we do not specify any relationship between C'_i and C_{X_i} .

2.2.1 Solution preserving decompositions

A decomposed system can be used to *preserve solutions* of a distributed constraint system: to obtain a global solution σ for the distributed constraint system \mathcal{S} one simply merges the individual solutions σ_i of the subsystems \mathcal{S}_i of a decomposed system $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$. In that case the decomposed system is said to be solution preserving if the merging of *each collection* of local solutions σ_i always results in a global solution σ :

DEFINITION 1. Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a distributed constraint system. Then the decomposed system

$$\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$$

is said to be a *solution-preserving decomposition* w.r.t. \mathcal{S} if it satisfies the following property:

$$\emptyset \subseteq Sol(\mathcal{S}_1) \sqcup Sol(\mathcal{S}_2) \sqcup \dots \sqcup Sol(\mathcal{S}_N) \subseteq Sol(\mathcal{S}).$$

\mathcal{S}' is said to be strictly solution preserving if the first inclusion is strict whenever $Sol(\mathcal{S}) \neq \emptyset$.⁵

EXAMPLE 1. Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a constraint system where $C = \{x_1 \wedge x_2, x_1 \vee x_3, x_1 \vee x_4\}$ is a set of Boolean constraints over $X = \{x_1, x_2, x_3, x_4\}$ and X is partitioned as $\{X_1 = \{x_1, x_2\}, X_2 = \{x_3, x_4\}\}$. The decomposition $\{\mathcal{S}_1, \mathcal{S}_2\}$ where $\mathcal{S}_1 = \langle \{x_1, x_2\}, D_1, \{x_1 \wedge x_2\} \rangle$ and $\mathcal{S}_2 = \langle \{x_3, x_4\}, D_2, \emptyset \rangle$ is a strictly solution preserving decomposition of \mathcal{S} : \mathcal{S}_1 has a unique solution $Sol(\mathcal{S}_1) = \{\{x_1 \leftarrow 1, x_2 \leftarrow 1\}\}$, while \mathcal{S}_2 has a “universal” solution set: $Sol(\mathcal{S}_2) = \{\{x_3 \leftarrow i, x_4 \leftarrow j\} : i, j \in \{0, 1\}\}$. Every solution in $Sol(\mathcal{S}_1) \sqcup Sol(\mathcal{S}_2)$ is a solution to \mathcal{S} , because x_1 as well as x_2 is assigned to 1 in any merge, thereby satisfying C . Hence, $\{\mathcal{S}_1, \mathcal{S}_2\}$ is strictly solution preserving.

Note that, in general, not every solution $\sigma \in Sol(\mathcal{S})$ will be obtainable by simply merging local solutions σ_i .

2.2.2 Consistency preserving decompositions

In distributed database applications, one typically distinguishes local constraints from global (integrity) constraints. Usually, in such applications, agents are free to add constraints to their set of local constraints as long as the resulting set remains consistent. The problem then is to ensure that local consistency ensures global consistency. This global consistency has to be ensured by the set of integrity constraints. In order to prevent communication overload between the distributed sites, one often tries to distribute these integrity constraints over the sites in such a way that satisfaction of all the local versions of the constraints implies the satisfaction of the global constraints. To simplify the discussion, we focus on the case where each site is allowed to *add* constraints to their local store. Consistency preservation then means that the total set of original constraints plus locally added constraints is consistent, whenever the added information does not cause any local inconsistency. We need the following definition.

DEFINITION 2. Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a distributed constraint system. An extension of $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ is a constraint system $\mathcal{S}^E = \langle \{X_i\}_{i=1}^N, D, C' \rangle$ where $C \subseteq C'$.

⁵This last condition is needed to take care for inconsistent constraint systems.

An extension \mathcal{S}^E captures the idea of a constraint system \mathcal{S} to which a set constraints has been added. Suppose that we have a decomposed system $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ and suppose that to each local system \mathcal{S}_i a set $C''_i \setminus C'_i$ of constraints is added such that for each \mathcal{S}_i we obtain an extension \mathcal{S}_i^E . Then consistency preservation requires that local consistency implies global consistency. That is, if the locally added information $C''_i \setminus C'_i$ does not render any resulting extension \mathcal{S}_i^E inconsistent, the total information $(C''_1 \setminus C'_1) \cup \dots \cup (C''_N \setminus C'_N)$ added to the distributed system should not render the total system inconsistent, i.e., $\mathcal{S}^E = \langle X, D, C \cup (C''_1 \setminus C'_1) \cup \dots \cup (C''_N \setminus C'_N) \rangle$ should be consistent as well.

DEFINITION 3 (CONSISTENCY PRESERVING EXTENSIONS). *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a distributed constraint system. A decomposition*

$$\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$$

is called consistency preserving w.r.t. \mathcal{S} if the following condition holds: whenever, for all $i = 1, 2, \dots, N$, the extensions

$$\mathcal{S}_i^E = \langle X_i, D_i, C''_i \rangle$$

of \mathcal{S}_i are consistent, the global extension

$$\mathcal{S}^E = \langle X, D, C \cup (C''_1 \setminus C'_1) \cup \dots \cup (C''_N \setminus C'_N) \rangle$$

is consistent as well. \mathcal{S}' is said to be strictly consistency preserving if, moreover, it holds that every \mathcal{S}_i is consistent whenever \mathcal{S} is consistent.

EXAMPLE 2. *Consider the distributed constraint system specified in Example 1 and its decomposition $\mathcal{S}' = \{\mathcal{S}_1, \mathcal{S}_2\}$. We show that \mathcal{S}' is also a strictly consistency preserving decomposition w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$: Every constraint over X_i added to the local constraint systems \mathcal{S}_i that keeps it consistent, will imply the existence of a non-empty set of solutions for \mathcal{S}_i . Since \mathcal{S}_1 has a unique solution, every consistent extension \mathcal{S}_1^E must have the same unique solution $\sigma_1 = \{x_1 \leftarrow 1, x_2 \leftarrow 1\}$. Whatever solution σ_2 is chosen for a consistent \mathcal{S}_2^E , it is always a solution to \mathcal{S}_2 , too. But then, using the solution preservation property, $\sigma = \sigma_1 \sqcup \sigma_2 \models C$. Moreover, $\sigma_1 \models C''_1$ and $\sigma_2 \models C''_2$, therefore*

$$\sigma \models C \cup (C''_1 \setminus C'_1) \cup (C''_2 \setminus C'_2).$$

Hence, \mathcal{S}^E is consistent and the decomposition is strictly consistency preserving.

2.2.3 Safety preserving decompositions

In areas such as P2P systems and sensor networks, one uses global constraints on the values by variables indicating vital system properties or to characterize the normal behavior of a system. As long as these global constraints are satisfied, no active control of the system is necessary. Only if violations of these global constraints occur, actions have to be performed to restore a normal state. In order to avoid excessive communication between the sites, one prefers to detect such anomalies in a distributed way. That is, the global constraints need to be localized in such a way that each site can establish independently from the others whether or not its local set of constraints is violated. Such a detection mechanism should be *safe* in the sense that whenever there is a global violation, at least one site should have detected it. But this, by contraposition, immediately implies that

safeness also can be expressed as a preservation property: whenever each local site concludes that its local set of constraints is safe, the global set of constraints should be safe, too.

DEFINITION 4 (SAFETY PRESERVING DECOMPOSITIONS). *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a distributed constraint system. A decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ is called safety preserving w.r.t. \mathcal{S} if the following condition holds: whenever there exists a global system state (assignment) σ such that for all $i = 1, 2, \dots, N$, \mathcal{S}_i is locally safe, i.e., $\sigma_i \models C'_i$, then the global system is safe, too, i.e., $\sigma \models C$.*

EXAMPLE 3. *Take $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ of Example 1. The decomposition $\mathcal{S}' = \{\mathcal{S}_1, \mathcal{S}_2\}$ is obviously safety preserving w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$: whenever there is a state σ such that $\sigma_1 \models (x_1 \wedge x_2)$ and $\sigma_2 \models \text{true}$, we must have that $\sigma \models (x_1 \wedge x_2) \wedge (x_1 \vee x_3) \wedge (x_1 \vee x_4)$.*

3. DECOMPOSITION PROPERTIES: RELATIONSHIPS

Given the three preservation properties we distinguished in decompositions of constraint systems, the first question we should answer is how they are related: Are they independent? Is one subsumed by the other? Or are they in fact equivalent? We start with the easiest one. As the reader might have noticed, there is an obvious relationship between solution preserving decompositions and safety preserving decompositions: A decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ is solution preserving exactly when it is safety preserving with respect to a given distributed system $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$:

PROPOSITION 1. *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a distributed constraint system. Then $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ is solution preserving w.r.t. \mathcal{S} iff \mathcal{S}' is also safety preserving w.r.t. \mathcal{S} .*

PROOF. Notice that a decomposition \mathcal{S}' that is safety preserving exactly if for all $\sigma = \sigma_1 \sqcup \dots \sqcup \sigma_N \in \Sigma$ it holds that $\forall i = 1, \dots, N$ $\sigma_i \in \text{Sol}(\mathcal{S}_i)$ implies $\sigma \in \text{Sol}(\mathcal{S})$.

Hence, \mathcal{S}' is safety preserving iff

$$\emptyset \neq \text{Sol}(\mathcal{S}_1) \sqcup \text{Sol}(\mathcal{S}_2) \cup \dots \cup \text{Sol}(\mathcal{S}_N) \subseteq \text{Sol}(\mathcal{S})$$

iff \mathcal{S}' is solution preserving. \square

With respect to consistency preserving and solution preserving decompositions, intuitively, it should be easy to show that solution preservation subsumes consistency preservation: if it is ensured that updates to a *local* constraint store ensure locally consistent stores, there exist local solutions σ_i for every updated local store. In particular, these solutions are solutions for the initial versions of the local stores. Hence, by solution preservation, merging these solutions constitutes a solution σ for the global (initial) store. But then it is easy to show that σ satisfies all the local updates as well. Hence, the global constraint store plus the added constraints is a consistent set as well. More precisely:

PROPOSITION 2. *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a distributed constraint system. If $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ is solution preserving w.r.t. \mathcal{S} , then \mathcal{S}' is also consistency preserving w.r.t. \mathcal{S} .*

PROOF. Assume \mathcal{S}' to be solution preserving w.r.t. \mathcal{S} . For $i = 1, 2, \dots, N$, consider arbitrary (consistent) extensions $\mathcal{S}_i^E = \langle X_i, D_i, C_i'' \rangle$ of the local subsystems $\mathcal{S}_i = \langle X_i, D_i, C_i' \rangle$. For each subsystem \mathcal{S}_i^E , select an arbitrary assignment $\sigma_i \in \text{Sol}(\mathcal{S}_i^E)$.

Since $C_i'' \supseteq C_i'$, it follows that

$$\emptyset \neq \text{Sol}(\mathcal{S}_i^E) \subseteq \text{Sol}(\mathcal{S}_i).$$

Hence, by solution preservation, the assignment $\sigma = \sigma_1 \sqcup \dots \sqcup \sigma_N$ will satisfy \mathcal{S} . Therefore,

$$\sigma \models C \quad (1)$$

By definition of σ_i , $\sigma_i \models C_i'' \setminus C_i'$. Moreover, every $C_i'' \setminus C_i'$ is a set of variable disjoint constraints over X_i . Hence, it follows that

$$\sigma \models (C_1'' \setminus C_1') \cup (C_2'' \setminus C_2') \cup \dots \cup (C_N'' \setminus C_N') \quad (2)$$

Hence, by equation (1) and (2),

$$\sigma \models C \cup (C_1'' \setminus C_1') \cup (C_2'' \setminus C_2') \cup \dots \cup (C_N'' \setminus C_N')$$

and therefore, $\sigma \in \text{Sol}(\mathcal{S}^E)$. So, $\text{Sol}(\mathcal{S}^E) \neq \emptyset$ and, consequently, \mathcal{S}' is consistency preserving with respect to $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$. \square

Although consistency preservation might seem to be a weaker property, somewhat surprisingly, the converse is also true: consistency preservation implies solution preservation. The intuition behind this result is that every *solution* to a constraint system can be *encoded* as a *special update* of the constraint store. The resulting constraint store will have this solution as its unique solution. By consistency preservation, the resulting global constraint store will be consistent. Hence, this decomposition will also be solution preserving, since the merge of all local solutions will be the unique solution of the resulting system. More formally:

PROPOSITION 3. *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a distributed constraint system. If $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i' \rangle\}_{i=1}^N$ is consistency preserving w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$, then \mathcal{S}' is also solution preserving w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$.*

PROOF. Assume $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i' \rangle\}_{i=1}^N$ to be consistency preserving w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$. By assumption, for every subsystem \mathcal{S}_i and every extension $\mathcal{S}_i^E = \langle X_i, D_i, C_i'' \rangle$ of \mathcal{S}_i , it must hold that, whenever the extended local systems \mathcal{S}_i^E are consistent, then the global extended system

$$\mathcal{S}^E = \langle X, D, C \cup (C_1'' - C_1') \cup \dots \cup (C_N'' - C_N') \rangle$$

is also consistent.

For each $i = 1, \dots, N$, let σ_i be an arbitrary solution to $\mathcal{S}_i = \langle X_i, D_i, C_i' \rangle$. Since $\{X_i\}_{i=1}^N$ is a partition, the assignment $\sigma = \sigma_1 \sqcup \dots \sqcup \sigma_N$ is well-defined. We have to show that $\sigma \in \text{Sol}(\mathcal{S})$.

For $i = 1, \dots, N$, consider the extensions $\mathcal{S}_i^E = \langle X_i, D_i, C_i'' \rangle$, where

$$C_i'' = C_i' \cup \{x = \sigma(x) : x \in X_i\}.$$

That is, each C_i' is extended with a set of unary constraints encoding the assignment $x \leftarrow \sigma(x)$ for every variable $x \in X_i$. Then, for every $i = 1, 2, \dots, N$, \mathcal{S}_i^E is consistent and each σ_i is the unique solution of \mathcal{S}_i^E .

By consistency preservation, the extension

$$\mathcal{S}^E = \langle X, D, C \cup (C_1'' \setminus C_1') \cup \dots \cup (C_N'' \setminus C_N') \rangle$$

is consistent, too. Hence $\text{Sol}(\mathcal{S}^E) \neq \emptyset$. Now observe that

$$C \cup (C_1'' \setminus C_1') \cup \dots \cup (C_N'' \setminus C_N') = C \cup \{x = \sigma(x) : x \in X\}$$

Hence, it follows that σ is the *unique solution* of \mathcal{S}^E and therefore, $\sigma \models C$. Hence $\sigma \in \text{Sol}(\mathcal{S})$ and the decomposition \mathcal{S}' is also solution preserving. \square

As an easy consequence of these propositions we have the following result:

THEOREM 1. *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a distributed constraint system. Then $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i' \rangle\}_{i=1}^N$ is solution preserving w.r.t. \mathcal{S} iff \mathcal{S}' is safety preserving w.r.t. \mathcal{S} iff \mathcal{S}' is consistency preserving w.r.t. \mathcal{S} .*

It is not difficult to show that these equivalences also hold for the strictly preserving versions. This immediately implies that all results that have been obtained for consistency preserving decompositions such as occur in [4, 12] can be used for solution preserving approaches to decomposition as well.

4. FINDING SOLUTION PRESERVING DECOMPOSITIONS

The equivalence between the three preservation properties of decompositions does not tell us how we could obtain such decompositions. In this section, we will discuss the problem of finding suitable decompositions. Given the above proven equivalences, in this section we concentrate on the solution preservation property of decompositions.

First, we prove the equivalence between our notion of solution preserving decompositions and the notion of *safe decompositions* as introduced by [4] for the purpose of consistency preserving decompositions. Then, using the definition of safe decompositions we show that deciding whether a decomposition \mathcal{S}' is solution preserving w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ in general is a coNP-complete problem.

Next, we prove that finding such a decomposition \mathcal{S}' is as hard (neglecting polynomial differences) as finding a solution for the original system \mathcal{S} .⁶

We start by defining the notion of a safe decomposition:

DEFINITION 5 ([4]). *Given a distributed constraint system $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$, the decomposition*

$$\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i' \rangle\}_{i=1}^N$$

is said to be a safe decomposition w.r.t. \mathcal{S} if

$$\bigcup_{i=1}^N C_i' \models C.$$

Note that this property is also sometimes known as the *covering property* [1] and should not be confused with the *safety preservation* property we discussed in the previous section.

PROPOSITION 4. *The decomposed system*

$$\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C_i' \rangle\}_{i=1}^N$$

is safe w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ iff \mathcal{S}' is solution preserving w.r.t. \mathcal{S} .

⁶Here, we assume that the class of allowable constraints always comprises the class of unary equality constraints of the form $x = d$ where $d \in \text{Dom}(x)$.

PROOF SKETCH. Assume that \mathcal{S}' is solution preserving w.r.t. \mathcal{S} . Then $Sol(\mathcal{S}_1) \sqcup \dots \sqcup Sol(\mathcal{S}_N) \subseteq Sol(\mathcal{S})$. Take an arbitrary assignment σ satisfying $\bigcup_{i=1}^n C'_i$. Then σ can be written as $\sigma = \sigma_1 \sqcup \sigma_2 \sqcup \dots \sqcup \sigma_N$, where $\sigma_i \models C'_i$ since $\{X_i\}_{i=1}^N$ is a partitioning. Therefore, for $i = 1, 2, \dots, N$, $\sigma_i \in Sol(\mathcal{S}_i)$. By solution preservation we have $\sigma \in Sol(\mathcal{S})$. Therefore, $\sigma \models C$ and the decomposition is safe w.r.t. \mathcal{S} .

Conversely, assume the decomposition \mathcal{S}' to be safe w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$. Then $\bigcup_{i=1}^n C'_i \models C$. So every assignment $\sigma : X \rightarrow D$ satisfying $\bigcup_{i=1}^n C'_i$ will also satisfy C . Each such a solution σ can be written as $\sigma = \sigma_1 \sqcup \sigma_2 \sqcup \dots \sqcup \sigma_N$ where each $\sigma_i : X_i \rightarrow D_i$ satisfies C'_i . Hence,

$$Sol(\mathcal{S}_1) \sqcup \dots \sqcup Sol(\mathcal{S}_N) \subseteq Sol(\mathcal{S})$$

and \mathcal{S}' is solution preserving w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$. \square

Using this notion of a safe decomposition, we can show that the problem of deciding whether a decomposition \mathcal{S}' is safe w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ is a coNP-complete problem:

PROPOSITION 5. *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a distributed constraint system and $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ be a decomposition. Then the problem to decide whether \mathcal{S}' is safe w.r.t. \mathcal{S} is coNP-complete.*

PROOF. To show that the problem is in coNP, just guess an assignment satisfying $\bigwedge_{i=1}^N C'_i$, but falsifying C . This shows the complement is in NP. coNP-hardness immediately follows from a reduction from the coNP-complete LOGICAL CONSEQUENCE problem: Given two propositional formulas ϕ and ψ , does it hold that $\phi \models \psi$. To see this, given arbitrary ϕ and ψ , let X_1 be the non-empty set of propositional atoms occurring in ϕ and ψ . Let $X_2 = \{y\}$ where y does not occur in X_2 . Consider the constraint system $\mathcal{S} = \langle X, D, C \rangle$ where $X = X_1 \cup X_2$, D is a set of Boolean domains and $C = \{\phi, \psi \vee y, \neg y\}$. Let $\mathcal{S}_1 = \langle X_1, D_{X_1}, \{\phi\} \rangle$ and $\mathcal{S}_2 = \langle X_2, D_{X_2}, \{\neg y\} \rangle$. Then $\mathcal{S}' = \{\mathcal{S}_1, \mathcal{S}_2\}$ is a safe decomposition w.r.t. $(\mathcal{S}, \{X_1, X_2\})$ iff $(\phi \wedge \neg y) \models \{\phi, \psi \vee y, \neg y\}$ iff $\phi \models \{\phi, \psi\}$ iff $\phi \models \psi$. \square

So, unless P=NP, it is hard to decide whether a decomposition is solution preserving. We can, however, obtain a more detailed result by relating the difficulty of finding a strictly solution preserving decomposition for a constraint system \mathcal{S} belonging to a class of constraint systems to the difficulty of finding a solution to \mathcal{S} :

PROPOSITION 6. *Let \mathcal{C} be an arbitrary class of constraint systems allowing at least equality constraints. Then there exists a polynomial algorithm to find a solution for constraint systems \mathcal{S} in \mathcal{C} iff there exists a polynomial algorithm that, given a constraint system $\mathcal{S} \in \mathcal{C}$ and an arbitrary partition $\{X_i\}_{i=1}^N$ of X , finds a strictly solution preserving decomposition w.r.t. $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$.*

PROOF. Suppose that there exists a polynomial algorithm A to find a solution for constraint systems in \mathcal{C} . We show how to construct a polynomial algorithm for finding a decomposition for an arbitrary partition of such a constraint system. Let $\mathcal{S} \in \mathcal{C}$ be constraint system and $\{X_i\}_{i=1}^N$ an arbitrary partitioning of X . To obtain a decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ of \mathcal{S} , first, using A , we compute a solution σ of \mathcal{S} . For every X_i , let

$$C_{\sigma_i} = \{x = d \mid x \leftarrow d \in \sigma, x \in X_i\}$$

be a set of unary constraints for variables in X_i directly obtained from σ . Then the subsystems $\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle$ are simply obtained by setting $C'_i = C_{X_i} \cup C_{\sigma_i}$. Note that each of these subsystems \mathcal{S}_i has a unique solution $\sigma_i = \{x \leftarrow d \in \sigma \mid x \in X_i\}$ and the merging of these solutions σ_i equals σ , i.e., a solution to the original system \mathcal{S} . Clearly, $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ is a solution preserving decomposition for \mathcal{S} that can be obtained in polynomial time.

Conversely, suppose we can find a strictly solution preserving decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ for a constraint system $\mathcal{S} \in \mathcal{C}$ w.r.t. any partitioning $\{X_i\}_{i=1}^N$ in polynomial time. We show how to obtain a solution σ of \mathcal{S} in polynomial time.⁷ Since the decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ can be obtained for *any* partitioning of X , we choose the partitioning $\{X_i\}_{i=1}^N$ where $X_i = \{x_i\}$ for $i = 1, 2, \dots, N$. Since the decomposition can be obtained in polynomial time, it follows that $\left| \bigcup_{i=1}^N C'_i \right|$ is polynomially bounded in the size of the input \mathcal{S} . Hence, the resulting decomposed subsystems \mathcal{S}_i each consist of a polynomially bounded set of *unary* constraints. It is well known that such constraint systems are solvable in polynomial time [6]. Therefore, in polynomial time for each subsystem \mathcal{S}_i an arbitrary value $d_i \in D_i$ for x_i can be obtained, satisfying all constraints. Let $\sigma_i = \{x_i \leftarrow d_i\}$ denote the solution obtained for \mathcal{S}_i . Since $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ is a solution preserving decomposition, the merging $\sigma = \sigma_1 \sqcup \sigma_2 \sqcup \dots \sqcup \sigma_N$ must be a solution of \mathcal{S} as well. Therefore, σ is a solution of \mathcal{S} , too. Hence, given a polynomial algorithm for achieving a solution preserving decomposition, we can construct a solution $\sigma \in Sol(\mathcal{S})$ in polynomial time. \square

Using this result and Theorem 1, we now may conclude:

THEOREM 2. *Finding a strictly {consistency, solution, safety}-preserving decomposition \mathcal{S}' for a distributed constraint system \mathcal{S} is, neglecting polynomial-time differences, as hard as finding a solution for \mathcal{S} .*

It is well-known that for general constraint systems, finding a solution is NP-hard [7]. This theorem explains why sometimes finding decompositions for a constraint system is easy: one should restrict one's attention to tractable constraint systems as STNs [10] or linear arithmetic constraints [4].

5. OPTIMAL SOLUTION PRESERVING DECOMPOSITIONS

Finding an arbitrary solution preserving decomposition for a given distributed constraint system might not always be sufficient. One important property we also should pay attention to is the amount of information that is preserved in determining a (solution preserving) decomposition. For example, taking a safety preserving decomposition, one would like to minimize false alarms, i.e., one would minimize those events where a local constraint is violated, but the global integrity constraint would still be satisfied.

Hence, the information loss due to the decomposition $\mathcal{S} = \{\mathcal{S}_i = \langle X_i, D_i, C_i \rangle\}_{i=1}^N$ can be defined as $Sol(\mathcal{S}) \setminus (Sol(\mathcal{S}_1) \sqcup \dots \sqcup Sol(\mathcal{S}_N))$: the set of solutions of the original system that cannot be obtained by merging the local solutions using the decomposition.

⁷The case where \mathcal{S} is inconsistent is easy and omitted, here.

Therefore, given a distributed constraint system $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$, we would like to call a solution preserving decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ an *optimal decomposition* if it minimizes $|\text{Sol}(\mathcal{S}) \setminus \text{Sol}(\mathcal{S}')|$ where $\text{Sol}(\mathcal{S}') = \text{Sol}(\mathcal{S}_1) \sqcup \dots \sqcup \text{Sol}(\mathcal{S}_N)$ is the set of solutions obtainable from the decomposed system.⁸

This optimality problem can be easily shown to be intractable, even if the underlying constraint system contains two variables and one (binary) constraint⁹, and finding a solution preserving decomposition is trivial:

PROPOSITION 7. *Let $\mathcal{S} = \langle \{X_i\}_{i=1}^N, D, C \rangle$ be a distributed constraint system where $|X| = 2$ and C contains only one binary constraint. Then the problem to find an optimal solution preserving decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ w.r.t. \mathcal{S} , $\{X_i\}_{i=1}^N$ is an NP-hard problem.*

PROOF. (Sketch) Consider the COMPLETE BIPARTITE SUBGRAPH problem: Given a bipartite graph $G = (V_1 \cup V_2, E)$ and a positive integer K , does there exist a complete bipartite subgraph (bi-clique) of order K in G ? This problem can be easily shown to be NP-complete by a reduction from the standard CLIQUE problem.

Let $G = (V_1 \cup V_2, E)$ be an instance of the NP-hard MAXIMUM COMPLETE BIPARTITE SUBGRAPH problem. We create an instance of the optimal decomposition problem as follows: Let $\mathcal{S} = (X, D, C)$ be a constraint system and let $\{X_1, X_2\}$ be a partitioning of $X = \{x_1, x_2\}$, where $X_1 = \{x_1\}$ and $X_2 = \{x_2\}$ and the domain of x_1 is $D_1 = V_1$ and the domain of x_2 is $D_2 = V_2$. C contains only one constraint R_E which consists of exactly those tuples that occur in E , that is $(v_1, v_2) \in R_E$ iff $\{v_1, v_2\} \in E$.

Finding a solution preserving optimal decomposition $\mathcal{S}' = \{\mathcal{S}_i = \langle X_i, D_i, C'_i \rangle\}_{i=1}^N$ for $(\mathcal{S}, \{X_1, X_2\})$ would imply that we have to find two subsystems $\mathcal{S}_1 = (\{x_1\}, \{V_1\}, C_1)$ and $\mathcal{S}_2 = (\{x_2\}, \{V_2\}, C_2)$, such that $C_1 \times C_2$ is a cardinality maximal subset of the tuples of R_E . Note that both C_1 and C_2 contain unary relations R_1 and R_2 respectively, where $R_1 = V'_1 \subseteq V_1$ and $R_2 = V'_2 \subseteq V_2$, respectively. Then $C_1 \times C_2$ is a cardinality maximal subset of the tuples of R_E iff $(V'_1 \cup V'_2, V'_1 \times V'_2)$ is a cardinality maximal complete bipartite subgraph of G . \square

Note that this result shows that finding optimal solution preserving decompositions can be hard even in cases finding a solution preserving decomposition is easy. Hence the intimate complexity connection between finding optimal solution preserving decompositions and finding solutions for the underlying constraint system has been lost.

6. CONCLUSIONS & FUTURE WORK

This paper considered decompositions of distributed constraint problems and studied the relationship between two well-known properties of such decompositions: solution preservation and consistency preservation. While in database applications one is interested in finding consistency preserving decompositions that allow for local updating, in multi-agent

⁸Here, we concentrate on the case that these solution sets are *finite*, e.g., by requiring the domains D_i to be finite. Note that the problem then is in $P^{\#P}$ and hardness for this class is still open.

⁹A binary constraint is a constraint in which only two variables do occur

systems applications, one looks for solution preserving decompositions that allow for easy composition of local solutions. In this paper, we showed formally that these preservation notions in decomposition are equivalent. Concentrating on solution preserving decompositions, we proved that there exists an intimate connection between finding solution preserving decompositions for a given constraint system \mathcal{S} and finding solutions for \mathcal{S} : they are computationally equally hard, neglecting polynomial differences. Finally, we discussed finding optimal decompositions and showed that this problem is NP-hard even for partitions having only two blocks. Moreover, the connections between finding optimal decompositions for a constraint system and finding solutions for it are lost.

We would like to point out the following implications: First of all, Hunsberger [10] showed the tractability of the decomposition method in the special case of Simple Temporal Networks (STNs); in particular he showed that there exists a polynomial algorithm for finding solution preserving decompositions. This result should not come as a surprise given the results we have shown above and the fact that finding a solution for STNs is solvable in polynomial time. Secondly, in [4] it is shown that a safe decomposition can be easily found in case the constraints are linear arithmetic constraints. Again, this result is a consequence of the relationship between finding decompositions of a system \mathcal{S} and finding solutions for it. Therefore, viewed in this broader perspective, these two results can be seen as consequences of more general results.

With respect to decomposition in distributed scheduling problems, solution preserving decomposition methods of the type we have discussed can be applied to enable autonomous distributed scheduling without the necessity to coordinate the integration of the solutions and to solve conflicts between the individual schedules. Our results also show that if these decompositions are strictly solution preserving, such a decomposition would also allow for adding local constraints while maintaining local consistency without endangering the feasibility of the joint schedule.

Furthermore, we should point out that the work on plan coordination by design [16, 5] is closely related to the current decomposition approach. This work on plan coordination allows a set of partially ordered tasks to be distributed among a set of agents in such a way that each of the agents is able to compose its own plan for the set of tasks assigned to it while guaranteeing that merging these independently constructed plans always will result in a feasible joint plan. This preservation property can be conceived as an acyclicity preservation property, since it guarantees that the joint plan always is acyclic whenever the local plans are. Instead of allowing all possible additions of constraints by the individual planning agents, the only constraints an agent is allowed to add are precedence order constraints between tasks assigned to the agent.

Note that there are other views on decomposition in constraint systems, as expressed by *structural decomposition methods* [8, 17, 6, 14] and by the *distributed constraint optimization (DCOP) approach* [19, 13]. In the structural decomposition view (i) the structure of the problem (i.e., the set of constraints) dictates the way in which the subproblems are generated and (ii) in general, the decomposition will not allow the subproblems to be *independently* solvable. In the DCOP approach, the partitioning of the vari-

ables is given, but, in general, the result of decomposition is not a set of independently solvable subproblems. Our approach differs from these approaches in the sense that, using the autonomous agent perspective, unlike the structural decomposition approach, we are interested in decomposition methods that take a *given* partitioning of the variables into account. Secondly, unlike the DCOP and structural decomposition approach, we require a *complete decomposition* of the original problem instance, that is, we would like to find a set of subproblems that can be solved *concurrently* and *independently* to obtain a complete solution to the original instance.

Concerning future work, we would like to point out that in distributed scheduling there are other important preservation properties like makespan or tardiness preservation in decompositions that can be studied. In [18] we have made a preliminary investigation into minimal makespan preserving decompositions of scheduling problems, but a systematic investigation of the correspondence between these and other preservation properties is still lacking.

Furthermore, we should investigate the idea of stratified decomposition in AI applications where first a solution preserving decomposition of the first layer of their constraints can be provided, the agents submit their own preferred solutions and conditional on these solutions the decomposition of the next layer is provided, etc. This would allow for some kind of synchronisations, for example, when exactly one of two variables needs to be true, but each are owned by different agents (this is a very common problem when assigning duties or tasks to agents).

Finally, there is another interesting extension of the current approach quite similar to the work of [3], where decomposition is restricted to local constraints and variables occurring in the global constraints might be subject to further negotiation between agents, or subject to a special decomposition approach after agents have had an opportunity to express their preferences for the values of these variables. In such a way we could make a distinction between those parts of a constraint network that can be solved by the agents independently from each other and those parts that would require some additional processing.

7. REFERENCES

- [1] S. Agrawal, S. Deb, K. V. M. Naidu, and R. Rastogi. Efficient detection of distributed constraint violations. In *Proceedings of the 23rd International Conference on Data Engineering, ICDE 2007, April 15-20, 2007, Istanbul, Turkey*, pages 1320–1324. IEEE, 2007.
- [2] A.A. Alwan, H. Ibrahim, and N. Izura Udzir. Improved integrity constraints checking in distributed databases by exploiting local checking. *Journal of Computer Science and Technology*, 24(4):665–674, 2009.
- [3] J.C. Boerkoel and E.H. Durfee. Partitioning the multiagent simple temporal problem for concurrency and privacy. In R. I. Brafman, H. Geffner, J. Hoffmann, and H.A. Kautz, editors, *Proceedings of the 29th International Conference on Automated Planning and Scheduling, ICAPS 2010*, pages 26–33. AAAI, 2010.
- [4] A. Brodsky, L. Kerschberg, and S. Varas. Optimal constraint decomposition for distributed databases. In M.J. Maher, editor, *Advances in Computer Science - ASIAN 2004*, volume 3321 of *Lecture Notes in Computer Science*, pages 301–319. Springer, 2004.
- [5] P.C. Buzing, A.W. ter Mors, J.M. Valk, and C. Witteveen. Coordinating self-interested planning agents. *Autonomous Agents and Multi-Agent Systems*, 12(2):199–218, March 2006.
- [6] D.A. Cohen, M. Gyssens, and P. Jeavons. A unifying theory of structural decompositions for the constraint satisfaction problems. In *Complexity of Constraints. Dagstuhl Seminar Proceedings 06401*, 2006.
- [7] R. Dechter. *Constraint Processing*. Morgan Kaufmann Publishers, 2003.
- [8] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural CSP decomposition methods. *Artificial Intelligence*, 124:2000, 1999.
- [9] A. Gupta and J. Widom. Local verification of global integrity constraints in distributed databases. *SIGMOD Rec.*, 22(2):49–58, 1993.
- [10] L. Hunsberger. Algorithms for a temporal decoupling problem in multi-agent planning. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence (AAAI-2002)*, pages 468–475, 2002.
- [11] M. Karimadini and H. Lin. Synchronized Task Decomposition for Cooperative Multi-agent Systems. *ArXiv e-prints, 0911.0231K*, November 2009.
- [12] S. Mazumbar and P.K. Chrysantis. Localization of integrity constraints in mobile databases and specification in PRO-MOTION. *Mobile Networks and Applications*, 9(5):481–490, 2004.
- [13] P.J. Modi, W.M. Shen, M. Tambe, and M. Yokoo. An asynchronous complete method for distributed constraint optimization. In *AAMAS '03: Proceedings of the second international joint conference on Autonomous agents and multiagent systems*, pages 161–168, New York, NY, USA, 2003. ACM.
- [14] W. Naanaa. A domain decomposition algorithm for constraint satisfaction. *J. Exp. Algorithmics*, 13:1.13–1.23, 2009.
- [15] M. Pietrzyk, S. Mazumdar, and R. Cline. Dynamic adjustment of localized constraints. *Lecture Notes in Computer Science*, pages 791–801, 1999.
- [16] A.W. ter Mors, C. Yadati, C. Witteveen, and Y. Zhang. Coordination by design and the price of autonomy. *Journal of Autonomous Agents and Multi-Agent Systems*, (on-line version, <http://dx.doi.org/10.1007/s10458-009-9086-9>), 2009.
- [17] B. W. Wah and Y. Chen. Constraint partitioning in penalty formulations for solving temporal planning problems. *Artificial Intelligence*, 170(3):187–231, March 2006.
- [18] C. Yadati, C. Witteveen, Y. Zhang, M. Wu, and H. La Poutré. Autonomous scheduling with unbounded and bounded agents. In Ralph Bergmann, Gabriela Lindemann, Stefan Kirn, and Michal Pechoucek, editors, *Multiagent System Technologies. 6th German Conference, MATES 2008*, Lecture Notes In Computer Science, pages 195–206. Springer -Verlag, 2008.
- [19] M. Yokoo and K. Hirayama. Distributed breakout algorithm for solving distributed constraint satisfaction problems. In *Proceedings of the Second International Conference on Multiagent Systems*, pages 401–408, 1996.

Decentralized Monitoring of Distributed Anytime Algorithms

Alan Carlin
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
acarlin@cs.umass.edu

Shlomo Zilberstein
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
shlomo@cs.umass.edu

ABSTRACT

Anytime algorithms allow a system to trade solution quality for computation time. In previous work, monitoring techniques have been developed to allow agents to stop the computation at the “right” time so as to optimize a given time-dependent utility function. However, these results apply only to the single-agent case. In this paper we analyze the problems that arise when several agents solve components of a larger problem, each using an anytime algorithm. Monitoring in this case is more challenging as each agent is uncertain about the progress made so far by the others. We develop a formal framework for decentralized monitoring, establish the complexity of several interesting variants of the problem, and propose solution techniques for each one. Finally, we show that the framework can be applied to decentralized flow and planning problems.

Categories and Subject Descriptors

I.2.8 [Problem Solving, Control Methods, and Search]

General Terms

Algorithms, Performance

Keywords

bounded rationality, meta-level control, Dec-MDP

1. INTRODUCTION

Anytime algorithms are algorithms that improve their solution as a function of time, and can return an answer when interrupted [7]. When such algorithms are used, one must decide when to stop computing and use the current solution. Russell and Wefald describe the situation: “first, real agents have only finite computational power; second, they don’t have all the time in the world... Typically, the utility of an action will be a decreasing function of time.” [16]. Several works have studied the tradeoff of utility for time in applied settings including intelligent system design [10], problem solving and search [19], and planning [21].

Much of the work so far has focused on a single decision maker, whereas work on bounded rationality in group decision making has been relatively sparse [6, 13]. To some extent, any approximate reasoning framework could be viewed as a form of bounded rationality. But unless one can establish some constraints on decision qual-

Cite as: Decentralized Monitoring of Distributed Anytime Algorithms, Alan Carlin and Shlomo Zilberstein, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 157-164.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

ity, such interpretations of bounded rationality are not very interesting. It seems more attractive to define bounded rationality as an optimization problem constrained by the availability of knowledge and computational resources. One successful approach is based on using decision-theoretic principles to monitor and control the base-level decision making procedure. It has been shown that this monitoring approach can be treated as a Markov Decision Process (MDP) and it can be solved optimally offline and used to optimize decision quality with negligible run-time overhead [9]. This approach to bounded rationality relies on *optimal metareasoning* [16]. That is, an agent is considered bounded rational if it monitors and controls its underlying decision making procedure optimally so as to maximize the comprehensive value of the decision. Additional formal approaches to bounded rationality have been proposed. For example, *bounded optimality* is based on a construction method that yields the best possible decision making program given a certain agent architecture [15]. The approach implies that a bounded rational agent will not be outperformed by any other agent running on the same architecture. This is a stronger guarantee than optimal metareasoning, but it is also harder to achieve.

Extending these computational models of bounded rationality to multi-agent settings is hard. Even if one assumes that the agents collaborate with each other – as we do in this paper – there is an added layer of complication. There is uncertainty about the progress that each agent makes with its local problem solving process. Thus the metareasoning process inherently involves non-trivial coordination among the agents. One existing approach for meta-level coordination involves multiple agents that schedule a series of tasks [14]. As new tasks arrive, each agent must decide whether to deliberate on the new information and whether to negotiate with other agents about the new schedule. Each agent uses an MDP framework to reason about its deliberation process. The coordination across agents is handled by negotiation, not by the MDP policy. A more recent work uses Dec-MDPs for meta-level control, with reinforcement learning to assign radars to agents [5].

In this paper, we extend optimal metareasoning techniques to collaborative multiagent systems. We consider a decentralized setting, where multiple agents are solving components of a larger problem by running multiple anytime problem solving algorithms concurrently. The main challenge is for each individual agent to decide when to stop deliberating and start taking action based on its own partial information. In some settings, agents may be able to communicate and reach a better joint decision, but such communication may not be free. We propose a formal model to study these questions and show that decentralized monitoring of anytime computation can be reduced to the problem of solving decentralized MDP (Dec-MDP) [3].

2. DECENTRALIZED MONITORING

We focus in this paper on a multiagent setting in which a group of agents is engaged in collaborative decision making. Each agent solves a component of the overall problem using an anytime algorithm – an algorithm that can be stopped at any time and provide an approximate solution to the problem. Solution quality increases with computation time according to some known probabilistic performance profile. The purpose of metareasoning is to monitor the progress of the anytime algorithms in a decentralized manner and decide when to stop deliberation.

Definition 1. The *decentralized monitoring problem* (DMP) is defined by a tuple $\langle Ag, Q, \vec{q}^0, A, T, P, U, C_L, C_G \rangle$ such that:

- Ag is a set of agents.
- Q is a set $Q_1 \times Q_2 \dots \times Q_n$, where Q_i is a set of discrete quality levels for agent i . At each step t , we denote the vector of agent qualities by $\vec{q}^t \in Q$, or more simply by $\vec{q} \in Q$, whose components are $q_i \in Q_i$. Components of \vec{q}^t are qualities for individual agents. We denote the quality for agent i at time t by q_i^t .
- $\vec{q}^0 \in Q$ is a joint quality at the initial step, known to all agents.
- $A = \{\text{continue, stop, monitorL, monitorG}\}$ is a set of meta-level actions available to each agent. The actions monitorL and monitorG represent locally and global monitoring, respectively.
- T is a finite horizon representing the maximum number of time steps in the problem.
- P_i is the transition model for the “continue” action for agent i . We will use notation P with i implied by context. For all $i, t \in \{0..T-2\}$, $q_i^t \in Q_i$, and $q_i^{t+1} \in Q_i$, $P(q_i^{t+1}|q_i^t) \in [0, 1]$. Furthermore, $\sum_{q_i^{t+1} \in Q_i} P(q_i^{t+1}|q_i^t) = 1$. We assume that the transitions of any two agents i and j are independent of each other, that is, $P(q_i^{t+1}|q_i^t, q_j^t) = P(q_i^{t+1}|q_i^t)$.
- $U(\vec{q}, t) : Q \times T \rightarrow \mathfrak{R}$ is a utility function that represents the value of solving the overall problem with quality vector \vec{q} at time t .
- C_L and C_G are the costs of the local monitoring and global monitoring actions respectively.

Each agent solves a component of the overall problem using an anytime algorithm. Unless a “stop” action is taken by one of the agents, all the agents continue to deliberate for up to $T-1$ time steps.

At each time step, agents decide which option to take, to continue, stop, or monitor globally or locally. If all the agents choose to continue, then the time step is incremented and solution quality transitions according to P . However, agents are unaware of the new quality state determined by the stochastic transition. If any agent chooses to “stop”, then all agents are instructed to cease computation before the next time step, and the utility $U(\vec{q}, t)$ of the current solution is taken as the final utility. If an agent chooses to monitor locally, then a cost of C_L is subtracted from the utility (for each agent that chooses monitorL) and the agent becomes aware of its local quality at the current time step. If any agent chooses to monitor globally, a single cost of C_G is subtracted from the utility and all agents become aware of all qualities at the time step. The time step is not incremented after a monitoring action. After an agent chooses to monitor, it must then choose whether to continue or stop, at the same time step.

Agents are assumed to know the initial quality vector \vec{q}^0 . An agent has no knowledge about quality in later time steps, unless a

monitoring action is taken. The “monitorL” action monitors the local quality; when agent i takes the “monitorL” action at time t it obtains the value of q_i^t . However, it still does not know any component of \vec{q}^t . A “monitorG” action results in communication among all the agents, after which they all obtain the global quality \vec{q}^t .

3. LOCAL MONITORING

In this section, we examine the concept of local monitoring. That is, each agent must decide whether to continue its anytime computation, stop immediately, or monitor its progress at a cost C_L , and then decide whether to continue or stop deliberation and initiate joint execution. The main result in this section shows that a DMP with local monitoring decisions can be solved by first converting the problem to a Transition Independent Dec-MDP. Although the termination decision may seem to imply transition dependence, the dependence is eliminated in the construction of Theorem 1.

3.1 Complexity of Local Monitoring

When $C_L = 0$, each agent should choose to monitor locally on every step, since doing so is free. When $C_G = \infty$, agents should never choose to monitor globally. The following lemma and theorem shows that even for the simpler case where $C_L = 0$, $C_G = \infty$, and the number of agents is fixed, the problem of finding a joint optimal policy is NP-complete. The termination decision alone, made by agents with local views of quality, is NP-hard.

LEMMA 1. *The problem of finding an optimal solution for a DMP with a fixed number of agents $|Ag|$, $C_L = 0$ and $C_G = \infty$ is NP-hard in the number of quality levels.*

PROOF. A nearly identical problem to this special-case DMP with zero monitoring cost is the Team Decision Problem (TDP) in Tsitsiklis [22]. Unfortunately, unlike in the Team Decision Problem, three joint decisions of a two-agent DMP (when either agent stops, or they both do) contain the same utility. Therefore we proceed directly to the underlying Decentralized Detection problem upon which the complexity proof of TDP is established.

We show that the NP-complete Decentralized Detection (DD) problem can be solved by a three step DMP. The following definition is provided in [22].

Decentralized Detection: Given finite sets Y_1, Y_2 , a rational probability mass function $p : Y_1 \times Y_2 \rightarrow Q$, a partition $\{A_0, A_1\}$ of $Y_1 \times Y_2$. The goal is to optimize $J(\gamma_1, \gamma_2)$ over the selection of $\gamma_i : Y_i \rightarrow \{0, 1\}$, $i = 1, 2$, where $J(\gamma_1, \gamma_2)$ is given by

$$\sum_{(y_1, y_2) \in A_0} p(y_1, y_2) \gamma_1(y_1) \gamma_2(y_2) + \sum_{(y_1, y_2) \in A_1} p(y_1, y_2) (1 - \gamma_1(y_1) \gamma_2(y_2))$$

Decentralized detection can be polynomially reduced to a three step DMP with $C_L = 0$. The first step is a known joint quality \vec{q}^0 . We define a unique quality level at the second and third step for each $y_i \in Y_i$. We will denote the quality level representing y_i by q_{y_i} . Transition probabilities to the second step are defined by the probability mass function, $P(q_i^2, q_j^2) = p(y_1, y_2)$. Each agent then monitors (for zero cost) and is aware of its local quality.

We model the decision of selecting $\gamma_i = 1$ as a decision by agent i to continue, and of selecting $\gamma_i = 0$ as a decision by agent i to terminate. To accomplish this, the DMP transition model transitions deterministically to a unique quality at step 3, for each quality of step 2 of each agent.

Utility on step 3 is defined so that:
 $U(q_{y_i}^2, q_{y_j}^2, 2) = 0, U(q_{y_i}^3, q_{y_j}^3, 3) = 1$ iff $(y_i, y_j) \in A_0$, and
 $U(q_{y_i}^2, q_{y_j}^2, 2) = 1, U(q_{y_i}^3, q_{y_j}^3, 3) = 0$ iff $(y_i, y_j) \in A_1$.

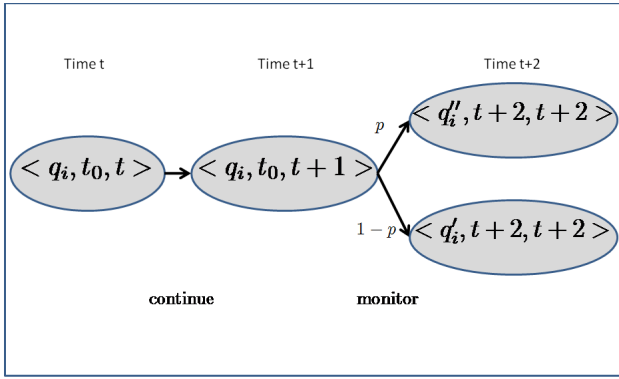


Figure 1: An example of the state space for one of the agents, while running the Dec-MDP construction of Theorem 1. When the agent continues, only the current time is incremented. When the agent monitors, the agent stochastically transitions to a new quality state based on its performance profile. The current time increments and monitoring time is set to the current time. Not shown, when either agent terminates, the agents get a reward based on the expectation of utility over their performance profiles.

It should be clear from this construction that an optimal continuation policy which maps q_{yi} to a decision to continue or terminate, can be used to construct $\gamma(y_i)$ in DD. \square

To show that this DMP is in NP, we will reduce to a transition independent Decentralized MDP (Dec-MDP), a problem which was shown by Goldman and Zilberstein to be NP-complete [8].

In the Dec-MDP model, each agent has a local state space (denoted S^i) available, similar to a classic MDP. The vector of states, one for each agent, is referred to as the joint state. Each agent takes one action from a set of actions denoted A^i , and actions have stochastic effects which change the local state. The vector of actions, one per agent, is referred to as the joint action. Finally, agents receive a joint reward (denoted $R(\vec{s}, \vec{a})$) for taking a joint action from a joint state. Execution takes place sequentially, a joint action is taken from a joint state, a joint reward is received, and the process repeats. In a finite horizon problem, there are T repetitions, with T being the horizon of the problem. An agent’s Dec-MDP policy is a mapping from its history of states and actions to a plan for future actions, and is denoted π_i .

Each agent is aware of its own local state and local action, but not necessarily the states and actions of the other agents at run-time. It is aware, however, of the other agents’ policies which were formed at planning time. The Dec-MDP model enforces the rule that the joint state is jointly fully observable. That is, between the agents, the whole state can be observed at every step. In typical formulations of Dec-MDP, this means each agent is aware of its local state but not the state of the other agents. In a transition independent Dec-MDP, state transitions of each agent are fully independent of each other, no agent can have an effect another agent’s local state. The only dependency between the agents is with respect to joint reward.

THEOREM 1. *The problem of finding an optimal solution for a DMP with a fixed number of agents, $C_L = k$ and $C_G = \infty$ is NP-complete.*

NP-hardness follows from above, with $k = 0$ as a special case. To show NP-completeness, we show that the problem can be reduced

to a transition independent Dec-MDP. Policies and policy-values for the DMP will correspond to policies and policy-values for the transition-independent Dec-MDP. The conversion is as follows:

The **state space** S^i for agent i are tuples $\langle q_i, t_0, t \rangle$, where q_i is a quality level (drawn from Q_i), t_0 is the time step at which that quality level was monitored, and t is the number of the current time step. We also define a terminal state for each agent.

The **action space** for all agents is $\{\text{terminate, continue, monitor}\}$.

The **transitions** consist of the following: when the action is to continue, t is merely incremented. When the action is to terminate, the agent transitions to the terminal state. Let $P_{\text{MDP}}(s'|s, a)$ be the transition function of the Dec-MDP. Let t_0 represent the time step when an agent last monitored. When the action is to monitor, we have $\forall q_i, q'_i \in Q_i$:

$$P_{\text{MDP}}(\langle q'_i, t'_0, t' \rangle | \langle q_i, t_0, t \rangle, \text{monitor}) = 0$$

$$\text{if } t' \neq t \text{ or } t' \neq t_0.$$

$$P_{\text{MDP}}(\langle q'_i, t'_0, t' \rangle | \langle q_i, t_0, t \rangle, \text{monitor}) = P(q'_i | q_i^{t_0})$$

$$\text{if } t' = t'_0 = t$$

The **Reward** is defined as zero if all agents choose to continue, as $-kC$ if k agents choose to monitor and none of the agents are in a terminal state, as $U(\vec{q})$ if one of the agents chooses to terminate and none of the agents are in a terminal state, and as zero if any agent is in a terminal state.

This reduction is polynomial, as the number of agent states in the Dec-MDP is $|Q_i|T^2$ and number of actions is 3. The representation is transition-independent, as the state of each agent does not affect the state of the other agents. Note that when one agent terminates, the other agents do not enter a terminal state, such a specification would violate transition independence. Rather, this notion, that no reward is accumulated once any agent has terminated, is captured by the reward function. No reward is received if any of the agents are in a terminal state. Since reward is only received when one of the agents enters the terminal state, reward is only received once, and the reward received by the Dec-MDP is the same as the utility received by the DMP.

Figure 1 shows a visual representation of the Dec-MDP reduction from a DMP with local monitoring costs. State consists of a tuple consisting of a quality level, the time at which the quality was monitored, and the current time. The “continue” action in the first step increments the current time. The “monitor” action increments the monitoring time to the current time, and probabilistically transitions quality according to the transition probability of the DMP applied to multiple steps.

An optimal policy for the Dec-MDP produces an optimal policy for the corresponding multi-agent anytime problem. Note that the uncertainty of quality present when an agent does not monitor is simulated in the MDP. Even though, in an MDP, an agent always knows its state, in this reduction the transition is not executed until the monitoring action is taken. Thus, even though an MDP has no local uncertainty, an agent does not “know” its quality until the monitor action is executed, and thus the local uncertainty of the multi-agent anytime problem is represented.

3.2 Solution Methods with Local Monitoring

3.2.1 Greedy Solution

We first build a solution that adapts the single-agent approach of Hansen and Zilberstein to the multi-agent case [9]. The adaption considers the other agents to be a part of the environment, and thus we name it the Greedy approach. Greedy computation does not take into account the actions of the other agents, we will initiate a

greedy computation by assuming that the other agents always continue, and that they will never monitor or terminate. We will then build upon this solution to develop a *nonmyopic solution*. For ease of explanation, we will describe the algorithm from a single agent's point of view. It should be assumed that each agent is executing this algorithm simultaneously.

Each agent begins by forming a performance profile for the other agents. We will use the term Pr as a probability function assuming only "continue" actions are taken, extending the transition model P over multiple steps. Furthermore we can derive performance profiles of multiple agents from the individual agents, using the independence of agent transitions. For example, in the two agent case we use $Pr(\vec{q})$ as shorthand for $Pr(q_i)Pr(q_j)$.

Definition 2. A dynamic local performance profile of an anytime algorithm, $Pr_i(q'_i|q_i, \Delta t)$, denotes the probability of agent i getting a solution of quality q' by continuing the algorithm for time interval Δt when the currently available solution has quality q .

Definition 3. A greedy estimate of expected value of computation (MEVC) for agent i at time t is:

$$MEVC(q_i^t, t, t + \Delta t) = \sum_{\vec{q}^t} \sum_{\vec{q}^{t+\Delta t}} Pr(\vec{q}^t | q_i^t, t) Pr(\vec{q}^{t+\Delta t} | \vec{q}^t, \Delta t) (U(\vec{q}^{t+\Delta t}, t + \Delta t) - U(\vec{q}^t, t))$$

The first probability is the expectation of the current global state, given the local state, and the second probability is the chance of transition. Thus, MEVC is the difference between the expected utility level after continuing for Δt more steps, versus the expected utility level at present. Both of these terms must be computed based on the performance profiles of the other agents, and thus the utilities are summed over all possible qualities achieved by the other agents. Cost of monitoring, C_L , is not included in the above definition. An agent making a decision must subtract this quantity outside the MEVC term.

For typical utility functions, the agent faces a choice as to whether to continue and achieve higher quality in a longer time, or to halt and receive the current quality with no additional time spent. A monitoring policy makes that decision.

Definition 4. A monitoring policy $\Pi(q_i, t)$ for agent i is a mapping from time step t and local quality level q_i to a decision whether to continue the algorithm or act on the currently available solution.

It is possible to construct a stopping rule by creating and optimizing a value function for each agent. First, create a new local-agent value function U_i such that

$$U_i(q_i, t) = \sum_{\vec{q}_{-i}^t} Pr(\vec{q}_{-i}^t) U(< q_i, \vec{q}_{-i} >, t)$$

Next, create a value function using dynamic programming, one step at a time:

$$V_i(q_i, t) = \max_d \begin{cases} \text{if } d = \text{stop:} \\ U_i(q_i, t), \\ \text{if } d = \text{continue:} \\ \sum_{q_i^{t+\Delta t}} Pr(q_i^{t+\Delta t} | q_i^t) V_i(q_i, t + \Delta t) \end{cases}$$

to determine the following policy:

$$\pi_i(q_i, t) = \operatorname{argmax}_d \begin{cases} \text{if } d = \text{stop:} \\ U_i(q_i, t), \\ \text{if } d = \text{continue:} \\ \sum_{q_i^{t+\Delta t}} Pr(q_i^{t+\Delta t} | q_i^t) V_i(q_i, t + \Delta t) \end{cases}$$

	$q_1^t = 1$	$q_1^t = 2$	$q_1^t = 3$
$q_2^t = 1$	-2	0	-1
$q_2^t = 2$	5	-3	-1
$q_2^t = 3$	-2	-1	1

Table 1: An example of a case where greedy termination policy produces a poor solution. Entries represent the expected utility of continuing for a step.

where Δt represents a single time step and d is a variable representing the decision. In the above, a stop action yields an expected utility over the qualities of the other agents. A continue action yields an expectation over joint qualities at future step $t + \Delta t$. The above definitions exclude the option of monitoring (thus incurring the costs C_L and C_G), the choices are merely whether to continue or act. Thus, we must define a cost-sensitive monitoring policy, which accounts for C_L and C_G .

Definition 5. A cost-sensitive monitoring policy, $\Pi_{i,c}(q_i, t)$, is a mapping from time step t , quality level q_i , and monitoring cost c into a monitoring decision $(\Delta t, m)$ such that Δt represents the additional amount of time to allocate to the anytime algorithm, and m is a binary variable that represents whether to monitor at the end of this time allocation or to stop without monitoring.

Thus, a cost-sensitive monitoring policy at each step chooses to either blindly continue, monitor, or terminate. It can be constructed using dynamic programming and the value function below. The agent chooses Δt , how many steps to continue blindly, as well as whether to stop or monitor after. If it stops, it receives expected utility, if it monitors it achieves value with a penalty of C_L .

$$V_c(q_i, t) = \max_{d, \Delta t} \begin{cases} \text{if } d = \text{stop:} \\ \sum_{q_i^{t+\Delta t}} Pr(q_i^{t+\Delta t} | q_i^t) U_i(q_i, t + \Delta t) \\ \text{if } d = \text{monitor:} \\ \sum_{q_i^{t+\Delta t}} Pr(q_i^{t+\Delta t} | q_i^t) V_c(q_i, t + \Delta t) - C_L \end{cases}$$

A greedy monitoring policy can thus be derived by applying dynamic programming over one agent. On initialization, such an algorithm assigns each quality level on the final step a value, based on its expected utility over possible qualities of the other agents. Then, working backwards, it finds the value of the previous step, which is the maximum over: (1) the expected utility over the possible qualities of the other agents (if it chooses to stop). (2) The expected utility of continuing (if it chooses to continue). An algorithm to find a cost-sensitive monitoring policy can similarly find the expectation over each time step with and without monitoring, and compare the difference to the cost of monitoring.

3.2.2 Solution Methods: Modeling the Other Agents

The greedy solution can be improved upon to coordinate policies among all the agents. To illustrate, examine Table 1. Each entry represents the expected joint utility of continuing (thus increasing utility but also time cost), minus the expected utility of stopping. Assume all entries have equal probability and monitoring cost is zero, and that the value of stopping immediately is zero, and thus the values shown represent only the value of continuing. Agent 1 would greedily decide to continue if it is in state $q_1^t = 1$ only, as that is the only column whose summation is positive. Agent 2 would greedily continue if it has achieved quality $q_2^t = 2$, as that is the only row whose summation is positive. However, this would

mean that the agents continue from all joint quality levels which are in bold. The sum of these levels is negative, and the agents would do better by selecting to terminate all the time!

We solve the DMP with $C_G = \infty$ optimally by leveraging the bilinear program approach of Petrik and Zilberstein to solving transition independent Dec-MDPs [11]. We first convert the problem to the transition independent Dec-MDP model described above. We prune “impossible” state-actions, for example we prune states where $t_0 > t$, as an agent can not have last monitored in the future. Then we convert the resulting problem into a bilinear program. A bilinear program can be described by the following objective function and constraints for the two-agent case (the framework is extensible beyond two agents if more agent-vectors are added):

$$\begin{aligned} & \text{maximize}_{x,y} r_1^T x + x^T R y + r_2 y \\ & \text{subject to } B_1 x = \alpha_1 \\ & \quad B_2 y = \alpha_2 \end{aligned}$$

In our bilinear formulation of a DMP, each component of the vector x represents a joint state-action pair for the first agent (similarly, each component of y represents a state-action for the second agent). Following the construction of Theorem 1, each component of x represents a tuple $\langle q_1^{t_0}, t_0, t, a \rangle$ where q_1 represents the last quality observed, t_0 represents the time at which it was observed, t represents the current time, and a represents a continue, monitor, or terminate. Thus, the length of x is $3|Q_1|T^2$ (assuming no pruning of impossible state-actions). Each entry represents the probability of that state and action occurring upon policy execution.

The vectors r_1 and r_2 are non-zero for entries corresponding to state-actions that have non-zero local reward, for agents 1 and 2 respectively. We set these vectors to zero, indicating no local reward.

The matrix R specifies joint rewards for joint actions, each entry corresponds to the joint reward of a single state-action in x and y . Thus, entries in R correspond to the joint utility $U(\vec{q}, t)$ of the row and column state, when any agent terminates. For entries where one agent monitors and the other agent terminates or continues, we adjust reward by $-C_L$. Otherwise, joint reward is 0.

For the constraints, α_1 and α_2 represent the initial state distributions, and B_1 and B_2 and correspond to the dual formation of the total expected reward MDP [12]. Intuitively, these constraints are very similar to the classic linear program formulation of maximum flow. Each constraint represents a state triple, and each constraint assures that the probability of transitioning to the state (which is the sum of state-actions that transition to it, weighted by their transition probabilities) matches the probability of taking the outgoing state-actions (which is the three state-actions corresponding to the state triple). A special case is the start quality, from which outgoing flow equals 1.

Bilinear programs, like their linear counterparts, can be solved through methods in the literature [11]. These techniques are beyond the scope of this paper, one technique is to alternately fix x and y policies and solve for the other as a linear program. Although bilinear problems are NP-complete in general, in practice performance depends on the number of non-zero entries in R .

4. GLOBAL MONITORING

Next, we examine the case where agents can communicate with each other (i.e., monitor globally). We will analyze the case where $C_L = 0$ and $C_G = k$, where k is a constant. For ease of description, we describe an on-line approach to communication. The online approach can be converted to an offline approach by anticipating all possible contingencies. We decide whether to communicate based on decision theory, agents compute Value of Information

(VoI), which is defined as follows:

$$VoI = V^*(q_i, t) - V_{sync}(q_i, t) - C_G$$

where V^* represents the expected utility after monitoring, V_{sync} represents expected utility without monitoring (see below), and C_G is cost of monitoring. In order to support the computation of V_{sync} and V^* , joint policies are produced at each communication point (or, for the offline algorithm, at all possible joint qualities). We define a helpful term $V^*(\vec{q}, t)$, (which decomposes into $V^*(q_i, \vec{q}_{-i}, t)$ to more clearly identify the local agent), which is the value of a joint policy after communication and discovery of joint quality \vec{q} , as computed through the methodology of the last section with $C_L = 0$. From the point of view of agent i , the value after communicating can then be viewed as an expectation over the quality of the other agents, based on their profiles.

$$V^*(q_i, t) = \sum_{\vec{q}_{-i}} Pr(\vec{q}_{-i}, t) V^*(q_i, \vec{q}_{-i}, t)$$

Similarly, V_{sync} is the value attached to quality q_i and not communicating. This was computed as part of the local monitoring problem at the last point of communication, we use the subscript “sync” to remind us that $V_{sync}(q_i, t)$ depends on the policies created and qualities observed at the last point of communication.

Non-myopic policies require each agent to make a decision as to whether to communicate or not at each step, resulting in the construction of a table resembling Table 1. We examined this table in a previous section when deciding whether to continue or stop. The table is used similarly for global monitoring, except the decision made by each agent is whether to communicate or not to communicate. Communication by either agent forces both agents to communicate and synchronize knowledge. Entries represent the joint state, and are incurred if *either* agent 1 decides to communicate from the row representing its quality, *or* agent 2 decides to communicate from the column representing its local knowledge.

This problem, of deciding whether to communicate after each step, is NP-complete as well. We will show this by reducing to a transition independent Dec-MDP-Comm-Sync [2]. A Dec-MDP-Comm-Sync is a transition independent Dec-MDP with an additional property: After each step, agents can decide whether to communicate or not to communicate. If they do not communicate, agents continue onto the next step as with a typical transition independent Dec-MDP. If any agent selects to communicate, then all agents learn the global state. However, a joint cost of C_G is assessed for performing the communication. Agents form joint plans at each time of communication. The portion of the joint plan formed by agent i after step t is denoted π_i^t .

THEOREM 2. *The DMP problem with $C_L = 0$ and C_G is a constant, is NP-complete.*

The proof of NP-hardness is similar to Lemma 1.

To show that the problem is in NP, we can reduce the problem to that of finding the solution of a Dec-MDP-Comm-Sync. We create the following Dec-MDP-Comm-Sync from a DMP with $C_L = 0$.

- S^i is the set $Q_i \cup \{f_i\}$ for agent i , where f_i is a new “terminal” state for agent i .
- $A^i = \{\text{continue, terminate}\}$; the joint action set is $\prod_i A^i$.
- The transition model:

$$\begin{aligned} P_{\text{MDP}}(q_i^{t+1} | q_i^t, \text{continue}) &= P(q_i^{t+1} | q_i^t) \\ P_{\text{MDP}}(q_i^{t_2} | q_i^{t_1}, \text{continue}) &= 0, \forall (t_2 \neq t_1 + 1) \\ P_{\text{MDP}}(f_i | q_i^t, \text{terminate}) &= 1, \forall q_i^t \in S^i \end{aligned}$$

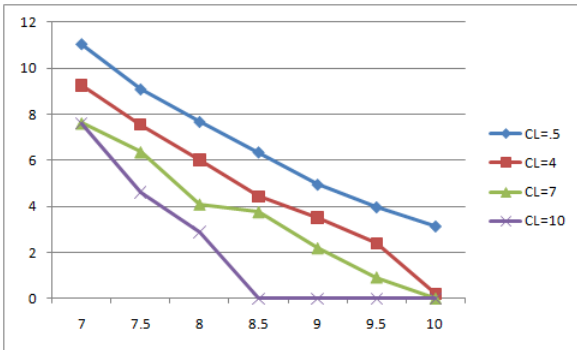


Figure 2: Expected Utility versus cost of time of non-myopic local monitoring of RockSample, for four different values of C_L . For the four plots, from highest to lowest, the cost of monitoring was .5, 4, 7, and 10.

- The reward function $R(\vec{q}^t, a_i) = U(\vec{q}, t)$ if $a_i = \text{terminate}$ for some i ; 0 otherwise.
- The reward function is 0 when any agent is in the final state.
- The horizon T is the same as T from the DMP.
- The cost of communication is C_G .

The reduction is polynomial as the number of states added is equal to T , and only one action is added.

It is straightforward to verify that this reduction is polynomial. Having represented the DMP problem as a Dec-MDP-Comm-Sync, we can use solution techniques from the literature to solve the problem which make use of a VoI computation [2].

5. EXPERIMENTS

We experimented on two decentralized decision problems involving anytime computation. First we profiled the RockSample domain, borrowed from the POMDP planning literature. In this planning problem, two rovers must each form a plan to sample rocks, maximizing the interesting samples. However, the location of the rocks are not known until runtime, and thus the plans can not be constructed until the rovers are deployed. We selected the HSVI algorithm for POMDPs as the planning tool [20]. HSVI is an anytime algorithm, the performance improves with time, its error bound is constructed and reported at runtime. Prior to runtime, the algorithm was simulated 10,000 times on randomized RockSample problems, in order to find the performance profile. The resulting profile held 5 quality levels over 6 time steps.

Second, we profiled the Ford Fulkerson maximum flow solution method. This motivating scenario involved a decentralized maximum flow problem where two entities must each solve a maximum flow problem in order to supply disparate goods to the customer. To estimate the transition model P in the DMP, we profiled performance of Ford Fulkerson through Monte Carlo simulation. The flow network was constructed randomly on each trial, with each edge capacity in the network drawn from a uniform distribution. Quality levels corresponded to regions containing equal-sized ranges of the current flow. From the simulation, a 3-dimensional probability table was created, with each layer of the table corresponding to the time, each row corresponding to a quality at that time, each column representing the quality at the next time step, and the entry representing the transition probability. We created software to compile a Decentralized MDP from the probability ma-

Problem (Local/Global)	Compile Time	Solve Time
Max Flow Local	3.5	11.4
RockSample Local	.13	2.8
Max Flow Global	.04	370
RockSample Global	.01	129

Table 2: Timing results in seconds. Compile time represents time to compile performance profile into bilinear problem, solve time measures time taken by the bilinear solver.

		Agent 1					
Quality		1	2	3	4	5	6
1		4	3	3	2	1	0
2		4	3	3	2	1	0
3		0	3	2	2	1	0
4		0	2	1	1	1	0
5		0	1	0	0	0	0

		Agent 2					
		1	2	3	4	5	6
1		4	3	3	2	1	0
2		4	3	3	2	1	0
3		0	3	2	2	1	0
4		0	2	1	1	1	0
5		0	1	0	0	0	0

Table 3: Greedy Local Monitoring Policy for RockSample problem, $C=2$ $K=8$

trix, as described in the previous sections, and solved the resulting problem using a bilinear program.

Three parameters of utility were varied with respect to each other: the reward for increasing quality, a linearly increasing cost of time, and the cost of monitoring. Experiments varied the latter two. For RockSample, the chosen utility function was $10(q_i + q_j - Kt)$, where t was time in seconds modulo 5, and we experimented with various values of K . For MaxFlow, utility was defined as $10(\min(q_i, q_j) - Kt)$, where $\min(q_i, q_j)$ represents the lesser of the flows, t represents the time step, (defined for the profile as the time in seconds modulo 5), and K represents cost of time and was varied. For max flow there were 10 quality levels and 10 time steps for each agent. In this section we focus problems with costs of time which result in non-trivial continuation policies. When cost of time is very low with respect to quality increase, computation trivially always continues, and when cost of time is very high, computation stops immediately. When cost of time lies between these two extremes, the option of monitoring becomes interesting.

Mean running time for the non-myopic variant of our algorithms is shown in Table 2. The MaxFlow problem was larger than the RockSample problem (containing more quality levels), thus consumed more time. The global formulations, as opposed to the local formulations, required a sub-problem formulation to precompute V^* at each communication point, and thus more time elapsed.

Table 3 shows resulting policies of the Greedy algorithm for local monitoring on the RockSample problem. The policy is taken for $K = 8$ and $C = 2$, which we consider well motivated since the routine to report progress took approximately a quarter of a time step. Rows represent quality levels and columns represent the time step at which that quality level was observed. Entries represent the agent's policy. Numbered entries represent to proceed that number of steps, and then terminate. For example, the entry in box (1, 1) represents that at step 1 with quality level 1, the Greedy policy proceeds 4 steps and then terminates. For this particular cost of time and cost of monitoring, the Greedy algorithm does not monitor at any step, although the Greedy algorithm did monitor for lower costs of monitoring. Also note that the Greedy algorithm is the same for agent 1 and agent 2, each is working off of the same profile and is

		Agent 1					
Quality		1	2	3	4	5	6
1		3M	-	-	1M	2	0
2		-	-	-	1M	2	0
3		-	-	-	1M	2	0
4		-	-	-	1M	1	0
5		-	-	-	0	0	0

		Agent 2					
		1M	1M	2M	-	1	0
1		1M	1M	2M	-	1	0
2		-	1M	2M	-	1	0
3		-	-	2M	-	1	0
4		-	-	1M	0	1	0
5		-	-	1M	0	0	0

Table 4: Nonmyopic local monitoring policy for first agent $C=2$ $K=8$. In some cases, the bilinear program returned stochastic actions, where multiple state-actions corresponding to the same state had non-zero probability (the sum of the state-actions equaled one). In these cases, the tables show the most probable action for the state.

not considering the policy of the other agent.

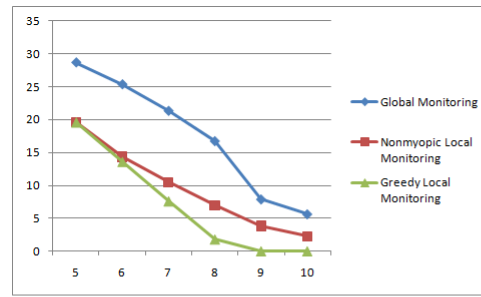
Table 4 shows resulting policies of the Nonmyopic algorithm for the same local monitoring problem. Entries designated “xM” denote continuing for x steps and then monitoring. For instance, in the first step, the first agent will proceed for 3 steps and then monitor. The second agent, by contrast, will proceed one step and then monitor. In these policies, agents are more likely to stop at higher quality levels, and more likely to monitor at earlier points in time.

Qualities that are impossible to achieve are reported with a “-”. The bilinear program only reports its policy for state-actions with probability above zero. For example, since agent 1 has a 100 percent chance of starting at quality level 1, and since the first step for agent 1 continues for three steps and then monitors, it is impossible to observe any quality level on the 2nd and 3rd steps.

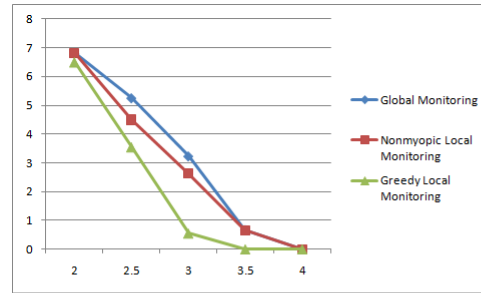
Figure 2 plots value versus the cost of time (K) for 4 different local costs of monitoring on the `RockSample` problem. For a constant cost of time, a higher cost of monitoring results in a lower quality solution. The drop off is monotonically decreasing and roughly linear, with higher cost of monitoring resulting in a more negative slope. This can be explained in context of the extremes of the graph. As one proceeds leftwards, cost of time is smaller, ultimately when it is small enough the agents should always continue (and, for instance when cost of time is zero, no monitoring decision is needed to verify this). As one proceeds rightwards, cost of time is larger, and when it is large enough the agents should stop on the first step (and again, no monitoring decision is required to verify this), achieving zero value. One can see that expected utility hits zero more quickly for the high cost of monitoring than the low cost of monitoring. Thus, the plots of all four monitoring costs intersect to the far left and the far right, but the zones where expected utility lies between those values is smallest when cost of monitoring is highest. Thus, higher cost of monitoring results in a more negative slope.

Figure 3 (a) shows value of Global Monitoring as compared to Local Monitoring on `RockSample` for various time costs. Value with Global Monitoring is higher, due to the Cost of Local Monitoring being zero for the Global Monitoring case, as well as the ability of each agent to monitor the progress of other agents, thus coordinating further.

Similarly, Figure 3 (b) summarizes experiments on `MaxFlow`. Nonmyopic monitoring outperforms myopic monitoring, and the global variant achieves higher performance. The `RockSample` domain proved more difficult to achieve higher scores, as the min function made it difficult to achieve value.



(a) `RockSample`



(b) `MaxFlow`

Figure 3: Comparison of global monitoring, nonmyopic local monitoring, and local monitoring on `RockSample` and `MaxFlow` problems.

6. RELATED WORK

The history of literature on anytime algorithms is rich in single-agent settings. We refer to [1, 6, 18] for recent overviews. Dean and Boddy used the term “anytime algorithm” in the 1980’s to describe a class of algorithms that “can be interrupted at any point during computation to return a result whose utility is a function of computation time” [7, 4]. They employed these algorithms in their work on time dependent planning and how to schedule deliberation. Horvitz, during the 1980’s as well, used decision theory to analyze “costs and benefits of applying alternative approximation procedures” to cases “where it is clear that there are insufficient computational resources to perform an analysis deemed as complete” [10]. Russell and Wefald used a discrete deliberation scheduling algorithm, which decides whether to deliberate or act based on expected value [16]. The work was implemented for search algorithms. Russell, Subramanian, and Parr utilized bounded optimality, which holds if a program produces a solution to a constrained optimization problem presented by the environment [15].

Work in artificial intelligence has produced several theories and architectures that can take into account the computational cost of decision making [7, 10, 15, 23, 24]. Zilberstein and Russell utilized performance profiles of algorithms in order to inform future anytime decisions [25]. The concept of performance profiles has been further explored in recent decision-theoretic approaches. Hansen and Zilberstein form a performance profile of an agent offline [9]. Then, based on this profile, a dynamic programming approach is used to make stopping decisions. The decisions use Bayesian reasoning based on the profiles in order to ascertain probability of future quality. Predictions of future quality are used to inform monitoring decisions, which are decisions whether to pause and monitor quality, or merely to continue. Similarly, Sandholm uses performance profiles to decide when to optimally terminate incomplete decision algorithms (algorithms which never finish if the answer is

N and may or may not finish if the answer is Y) on problems such as 3-SAT [17]. Termination decisions are based on the prior probability of an answer, on a utility model based on the utility of quality and time, and on the performance profiles.

In the multi-agent realm, Raja and Lesser explore a framework for coordinating agents Meta-Level control [13, 14]. In these works a single agent or multiple agents schedule a series of tasks. At various points in time, new tasks arrive, and each agent must decide whether to deliberate on the new information and whether to negotiate with other agents about the new schedule. The authors use an MDP framework within agents to reason about deliberation and the coordination across agents is handled by negotiation. The recent approach of Cheng et al. uses reinforcement learning for meta-level control of weather radars, using Dec-MDPs [5].

7. CONCLUSIONS AND FUTURE WORK

Anytime algorithms effectively gauge the trade-off between time and quality. Monitoring is an essential part of the process. Existing techniques from the literature weigh the trade-off between time, quality, and monitoring for the single-agent case. The complexity of the monitoring problem is known, and dynamic programming methods provide an efficient solution method.

However, this paper shows that these techniques do not scale to the multi-agent case. In this paper, we took a decision-theoretic approach to the monitoring problem. We formalized the problem for the multi-agent case, and proved that there exist problems for both local and global monitoring which are NP-complete. We showed how the multi-agent monitoring problems can be compiled as special cases of Decentralized Markov Decision Processes, and thus solvers from the literature can produce efficient solutions.

Future work lies in several directions. First, we will analyze and produce solutions for monitoring problems that are partially observable. We will also examine items like varying the monitoring cost. Second, we would like to examine cases with non-cooperative utility functions. Third, we will apply the methods to cases involving more than two agents. The latter will require modifications to the bilinear solver.

Acknowledgments

Support for this work was provided in part by the National Science Foundation Grant IIS-0812149 and by the Air Force Office of Scientific Research Grant FA9550-08-1-0181.

8. REFERENCES

- [1] M. Anderson. A review of recent research in metareasoning and metalearning. *AI Magazine*, 28(1):7–16, 2007.
- [2] R. Becker, A. Carlin, V. Lesser, and S. Zilberstein. Analyzing myopic approaches for multi-agent communication. *Computational Intelligence*, 25(1):31–50, 2009.
- [3] D.S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(1):819–840, 2002.
- [4] M. Boddy and T. Dean. Deliberation scheduling for problem solving in time-constrained environments. *Artificial Intelligence*, 67:244–285, 1994.
- [5] S. Cheng, A. Raja, and V. Lesser. Multiagent Meta-level Control for a Network of Weather Radars. In *Proceedings of 2010 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, 157-164, 2010.
- [6] M. Cox and A. Raja. *Metareasoning: Thinking about thinking*. Cambridge, MA: MIT Press.
- [7] T. Dean and M. Boddy. An analysis of time-dependent planning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 49–54, 1988.
- [8] C. Goldman and S. Zilberstein. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research*, 22:143–174, 2004.
- [9] E. Hansen and S. Zilberstein. Monitoring and control of anytime algorithms: A dynamic programming approach. *Artificial Intelligence*, 126(1-2):139–157, 2001.
- [10] E. Horvitz. Reasoning about beliefs and actions under computational resource constraints. In *Proceedings of Third Workshop on Uncertainty in Artificial Intelligence*, 429–444, 1987.
- [11] M. Petrik and S. Zilberstein. A bilinear approach for multiagent planning. *Journal of Artificial Intelligence Research*, 35:235–274, 2009.
- [12] M. Puterman. *Markov decision processes, Discrete stochastic dynamic programming*. John Wiley and Sons, Inc., 2005.
- [13] A. Raja and V. Lesser. Meta-level reasoning in deliberative agents. In *Proceedings of the International Conference on Intelligent Agent Technology*, 141–147, 2004.
- [14] A. Raja and V. Lesser. A framework for meta-level control in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 15:147–196, 2007.
- [15] S. Russell, D. Subramanian, and R. Parr. Provably bounded optimal agents. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 575–609, 1993.
- [16] S. Russell and E. Wefald. Principles of metareasoning. In *Proceedings of the First International Conference on Principles of Knowledge Representation and Reasoning*, 400–411, 1989.
- [17] T. Sandholm. Terminating decision algorithms optimally. In *Proceedings of the International Conference on Principles and Practice of Constraint Programming*, 2003.
- [18] M. Schut and M. Wooldridge. The control of reasoning in resource-bounded agents. *Knowledge Engineering Review*, 16(3):215–240, 2001.
- [19] H. Simon and J. Kadane. Optimal problem solving search: All or nothing solutions. *Computer Science Technical Report CMU-CS-74-41*, 1974.
- [20] T. Smith and R. Simmons. Heuristic search value iteration for POMDPs. In *Proceedings of the International Conference on Uncertainty in Artificial Intelligence*, 520–527, 2004.
- [21] M. Stefik. Planning and meta-planning. *Artificial Intelligence*, 16(2):141–170, 1981.
- [22] J. Tsitsiklis and M. Athans. On the complexity of decentralized decision making and detection problems. *IEEE Transactions on Automatic Control*, 30(5):440–446, 1985.
- [23] M. P. Wellman. *Formulation of Tradeoffs in Planning under Uncertainty*. London: Pitman, 1990.
- [24] S. Zilberstein. Operational rationality through compilation of anytime algorithms. Ph.D. Dissertation, Computer Science Division, University of California, Berkeley, 1993.
- [25] S. Zilberstein and S. Russell. Optimal composition of real-time systems. *Artificial Intelligence*, 82(1-2):181–213, 1996.
- [26] S. Zilberstein. Metareasoning and bounded rationality. In M. Cox and A. Raja (Eds.), *Metareasoning: Thinking about Thinking*, MIT Press, forthcoming.

Consensus Acceleration in Multiagent Systems with the Chebyshev Semi-Iterative Method

R. L. G. Cavalcante
Fraunhofer Institute for Telecommunications,
Heinrich Hertz Institute /
Technische Universität Berlin
renato.cavalcante@hhi.fraunhofer.de

A. Rogers, N. R. Jennings
University of Southampton
School of Electronics and Computer Science
{acr,nrj}@ecs.soton.ac.uk

ABSTRACT

We consider the fundamental problem of reaching consensus in multiagent systems. To date, the consensus problem has been typically solved with decentralized algorithms based on graph Laplacians. However, the convergence of these algorithms is often too slow for many important multiagent applications, and thus they are increasingly being combined with acceleration methods. Unfortunately, state-of-the-art acceleration techniques require parameters that can be optimally selected only if complete information about the network topology is available, which is rarely the case in practice. We address this limitation by deriving two novel acceleration methods that can deliver good performance even if little information about the network is available. The first is based on the Chebyshev semi-iterative method and maximizes the worst-case convergence speed given that only rough bounds on the extremal eigenvalues of the network matrix are available. It can be applied to systems where agents use unreliable communication links, and its computational complexity is similar to those of simple Laplacian-based methods. This algorithm requires synchronization among agents, so we also propose an asynchronous version that approximates the output of the synchronous algorithm. Mathematical analysis and numerical simulations show that the convergence speed of the proposed acceleration methods decrease gracefully in scenarios where the sole use of Laplacian-based methods is known to be impractical.

Categories and Subject Descriptors

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

General Terms

Algorithms, Theory

Keywords

Decentralized control, collective dynamics, consensus

1. INTRODUCTION

Reaching agreement (or consensus) between physically distributed agents is one of the fundamental requirements of many multiagent applications including target localization [1], distributed

Cite as: Consensus Acceleration in Multiagent Systems with the Chebyshev Semi-Iterative Method, R. L. G. Cavalcante, A. Rogers, and N. R. Jennings, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 165-172.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

coordination of large swarms of robots [2], distributed control of modular robotic actuators [3], and others [4]. Typically, such approaches require the agents to update a local estimate of an environmental or control parameter, by iteratively communicating with a few local neighbors, such that the estimates of all agents converge to the same value. For example, in [3] a modular robotic setting is described in which agents controlling decentralized modular actuators must reach consensus on the height of their actuators in order to ensure that a platform is kept level. Each agent can only infer (indirectly) the height of its neighbors, so only local control laws can be used, and the agents must reliably converge to a consensus height. Likewise, in one of the steps of the algorithm in [1], agents with individual estimates of the location of a target, iteratively exchange and update these estimates, with the intent that the estimates of all the agents converge to that which would have been reached had they been able to report their initial estimate to a center which could fuse them by taking their average.

To date, consensus problems of the type described above, have typically been solved with classic decentralized iterative algorithms based on graph Laplacians [2–5] and other related techniques that differ in the choice of the network matrices [6, 7]. In these consensus algorithms, every agent produces a sequence of estimates using a simple two-step approach that can be briefly described as follows. First, agents exchange estimates locally with their neighbors. Then each agent updates its current estimate by taking a weighted average of all estimates to which it has access, and the process repeats. As described above, the intent is that the estimates of all the agents converge to that which would have been reached had the average of the agents' initial estimates been taken. Unfortunately, however, it has been recently shown that the convergence of these classic iterative algorithms is often too slow in applications where agents have to agree on initial estimates that have a strong correlation with the agents' positions [2]. Typical scenarios in which this occurs are sensor networks and robotic swarms because the phenomena being measured are often functions of the agents' positions. In more detail, when the initial estimates are spatially correlated, the number of iterations required by Laplacian-based methods to compute the average consensus value with good accuracy can grow proportionally with the square of the network diameter [2]. This fact renders such methods impractical in large scale multiagent systems with sparse communication.

The convergence of these Laplacian-based algorithms can be greatly improved with acceleration techniques that filter the output [8–10]. In particular, efficient two-tap filters have been proposed for systems where agents communicate both synchronously [10] and asynchronously [8]. However, these algorithms typically have a free parameter that has to be chosen by the agents. Such heuristic choices of parameters can be avoided with the optimal polynomial

filtering approach proposed in [9]. Unfortunately, this approach requires precise knowledge of the mean value of the network matrix, which again is unlikely to be available in many multiagent systems. In addition, this method is only stable in systems where the communication links are fairly reliable.

Thus, to address the above shortcomings and to make Laplacian-based methods practical in the multiagent scenarios described earlier, we propose low-complexity acceleration methods based on digital filters that require little information about the network topology and are robust against unreliable communication links. In more detail, the main contributions of this study are as follows:

- We derive a novel acceleration method named *synchronous semi-iterative consensus*. This algorithm filters the output of classic consensus algorithms with a polynomial filter that is optimal in the sense of maximizing the worst-case mean convergence speed when the network topology is unknown. Unlike recent acceleration techniques [9], the proposed algorithm only requires rough upper and lower bounds on the extremal eigenvalues of the network matrix (first order statistics is not necessary), and it is amenable to an efficient recursive implementation. Compared to other state-of-the-art acceleration techniques, such as those in [8, 10], our synchronous semi-iterative consensus algorithm has better convergence properties in many practical scenarios, the same communication requirements, and roughly the same computational complexity.

- To handle scenarios where synchronization among agents is not possible, we further extend our approach to devise an asynchronous algorithm, named *asynchronous semi-iterative consensus*, that approximates the output of the proposed synchronous algorithm. This asynchronous algorithm has a strong connection with those in [8, 10], but it does not require heuristics for parameter tuning in real applications where the network topology is largely unknown. All parameters of our algorithm are readily obtained from rough upper and lower bounds of the extremal eigenvalues of the unknown network matrix.

The paper is divided as follows. Sect. 2 reviews classic consensus algorithms based on graph Laplacians and other similar approaches. Sect. 3 shows the two novel acceleration schemes. Numerical simulations in Sect. 4 evaluate the proposed methods in scenarios where Laplacian-based methods are impractical.

2. PROBLEM STATEMENT

We start by briefly introducing our notation. In particular, vectors are written in lower-case, bold typeface, and matrices are written in upper-case, bold typeface. Unless otherwise stated, vectors are assumed to be column vectors. For every vector $\mathbf{v} \in \mathbb{R}^N$, we define the norm of \mathbf{v} by $\|\mathbf{v}\| := \sqrt{\mathbf{v}^T \mathbf{v}}$, where $(\cdot)^T$ denotes the transpose operation. The vector of ones is denoted by $\mathbf{1}$, and its dimension is clear from the specific context. The element of the k th row and the j th column of a matrix $\mathbf{X} \in \mathbb{R}^{M \times N}$ is denoted by $[\mathbf{X}]_{kj}$. The eigenvalues of a symmetric matrix $\mathbf{X} \in \mathbb{R}^{N \times N}$ are denoted by $\lambda_1(\mathbf{X}), \dots, \lambda_N(\mathbf{X})$. By $\mathbf{D} := \text{diag}(\lambda_1, \dots, \lambda_N)$, we denote a diagonal matrix $\mathbf{D} \in \mathbb{R}^{N \times N}$ having $\lambda_1, \dots, \lambda_N$ as the entries on its main diagonal.

We now turn to the problem formulation. In this study we assume that the multiagent system forms a network represented by a connected undirected graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N} = \{1, \dots, N\}$ is the set of agents, $\mathcal{E} \subset \{\{k, j\} \mid k, j \in \mathcal{N}\}$ is the edge set, and the edge $\{k, j\} \in \mathcal{E}$ is an unordered pair of agents. For convenience, here we assume that $\{k, k\} \in \mathcal{E}$. Initially, at time $i = 0$, each agent k reports a value $x_k[0] \in \mathbb{R}$, and we are interested in iterative algorithms that produce, in every agent $k \in \mathcal{N}$, sequences $\{x_k[i]\}$ converging to $x_{av} := 1/N \sum_{k \in \mathcal{N}} x_k[0]$,

the average of the initial values reported by the agents.

To be truly decentralized, the algorithms of interest should respect the network topology, i.e., at time instant $i \in \mathbb{N}$, each agent k should exchange information only with its neighbors $\mathcal{N}_k := \{j \in \mathcal{N} \mid \{j, k\} \in \mathcal{E}\}$. In particular, classic algorithms having this desired feature take the form:

$$x_k[i+1] = \sum_{j \in \mathcal{N}_k} [\mathbf{W}[i]]_{kj} x_j[i], \quad k \in \mathcal{N}, \quad i \in \mathbb{N}, \quad (1)$$

or, more compactly,

$$\mathbf{x}[i+1] = \mathbf{W}[i] \mathbf{x}[i], \quad i \in \mathbb{N}, \quad (2)$$

where $\mathbf{x}[i] := [x_1[i] \dots x_N[i]]^T \in \mathbb{R}^N$, $\mathbf{W}[i] \in \mathbb{R}^{N \times N}$ is a properly selected sequence of (symmetric) matrices, $[\mathbf{W}[i]]_{kj}$ is the weight associated with the edge $\{k, j\}$ at time i , and $[\mathbf{W}[i]]_{kj} = 0$ if $\{i, j\} \notin \mathcal{E}$. To reach consensus, agents can compute the weights $[\mathbf{W}[i]]_{kj}$ in many different ways according to the desired characteristics of the system. In particular, if the network is deterministic, agents only know the local topology, and links are reliable, agents can use simple Laplacian-based methods to compute locally the weights [2, 4]. When links are unreliable, weights can be computed with the method in [5]. In systems where the network topology is known before deployment and links are deterministic, the approach in [6] can be used to compute a fixed matrix $\mathbf{W} = \mathbf{W}[i]$ that gives better convergence than simple heuristics based on graph Laplacians. In systems where agents operate asynchronously and do not know their neighbors, gossip consensus algorithms [7] can be used to determine the weights.

Hereafter, we do not use a specific method to compute the weights $[\mathbf{W}[i]]_{kj}$, and, for maximum generality, we only assume that the matrices $\mathbf{W}[i]$ satisfy the following properties:

ASSUMPTION 1. (*Properties of $\mathbf{W}[i]$*)

1. The matrices $\mathbf{W}[i]$ ($i \in \mathbb{N}$) in (2) are i.i.d. random¹ matrices with $[\mathbf{W}[i]]_{jk} = 0$ if $\{j, k\} \notin \mathcal{E}$.
2. Each matrix $\mathbf{W}[i]$ is symmetric and satisfies $\mathbf{W}[i] \mathbf{1} = \mathbf{1}$ (hence $\mathbf{W}[i]$ is a doubly stochastic matrix).
3. $\|E[\mathbf{W}[i]^T \mathbf{W}[i]] - 1/N \mathbf{1} \mathbf{1}^T\|_2 < 1$ (and $\|\overline{\mathbf{W}} - 1/N \mathbf{1} \mathbf{1}^T\|_2 < 1$, where $\overline{\mathbf{W}} := E[\mathbf{W}[i]]$ denotes the mean of $\mathbf{W}[i]$).

The above properties are sufficient conditions to guarantee that $\mathbf{x}[i]$ in (2) converges to $x_{av} \mathbf{1} \in \mathbb{R}^N$ in both the mean sense and the mean square sense [7], i.e., $\lim_{i \rightarrow \infty} E[\mathbf{x}[i]] = x_{av} \mathbf{1}$ and $\lim_{i \rightarrow \infty} E[\|\mathbf{x}[i] - x_{av} \mathbf{1}\|^2] = 0$. Unfortunately, irrespective of the method being used for the computation of the weights $[\mathbf{W}[i]]_{kj}$, when agents only have local information about the network topology, consensus algorithms solely based on the iteration in (2) are typically slow. In particular, when the initial values reported by agents have a strong correlation with their locations, Laplacian-based methods have been shown to be impractical in large multiagent systems because the convergence speed scales badly with the network diameter [2]. To address this serious drawback of consensus algorithms based on (2), we develop an acceleration technique that improves the convergence of $\mathbf{x}[i]$ in the mean sense. Before deriving the proposed method, we first review convergence properties of (2).

By the i.i.d. assumption of the symmetric matrices $\mathbf{W}[i]$ ($i \in \mathbb{N}$), we have that $E[\mathbf{W}[i]]$ is a time-invariant symmetric matrix ($E[\mathbf{W}[i]] = \overline{\mathbf{W}}$ for all $i \in \mathbb{N}$). Let the eigenvalue decomposition

¹In this study, we use the same notation for random variables and their realizations. The interpretation that should be applied is clear from the context.

of $\overline{\mathbf{W}}$ be given by $\mathbf{Q}\Lambda\mathbf{Q}^T = \overline{\mathbf{W}}$, where $\mathbf{q}_1, \dots, \mathbf{q}_N$ are the columns of \mathbf{Q} , and $\Lambda := \text{diag}(\lambda_1(\overline{\mathbf{W}}), \dots, \lambda_N(\overline{\mathbf{W}}))$ with eigenvalues arranged in non-increasing order of magnitude: $|\lambda_1(\overline{\mathbf{W}})| \geq \dots \geq |\lambda_N(\overline{\mathbf{W}})|$. Note that, also from Assumption 1, we have that $\mathbf{q}_1 = (1/\sqrt{N})\mathbf{1}$ and that $1 = \lambda_1(\overline{\mathbf{W}}) > |\lambda_j(\overline{\mathbf{W}})|$ for $j = 2, \dots, N$. With these definitions, we deduce:

$$\begin{aligned} E[\mathbf{x}[i]] &= (\overline{\mathbf{W}})^i \mathbf{x}[0] = \mathbf{Q}\Lambda^i \mathbf{Q}^T \mathbf{x}[0] \\ &= [(1/\sqrt{N})\mathbf{1} \ \mathbf{q}_2 \ \dots \ \mathbf{q}_N] \\ &\quad \cdot \text{diag}(1, (\lambda_2(\overline{\mathbf{W}}))^i, \dots, (\lambda_N(\overline{\mathbf{W}}))^i) \\ &\quad \cdot [(1/\sqrt{N})\mathbf{1} \ \mathbf{q}_2 \ \dots \ \mathbf{q}_N]^T \mathbf{x}[0]. \end{aligned} \quad (3)$$

Therefore, by $|\lambda_j(\overline{\mathbf{W}})| < 1$ for $j = 2, \dots, N$, we conclude that $\lim_{i \rightarrow \infty} E[\mathbf{x}[i]] = (1/N)\mathbf{1}\mathbf{1}^T \mathbf{x}[0] = x_{\text{av}}\mathbf{1}$ and that the slowest mode of convergence of (3) is given by $\lambda_2(\overline{\mathbf{W}})$ (by taking powers, the eigenvalues $\lambda_j(\overline{\mathbf{W}})$ ($j = 3, \dots, N$) do not decay slower to zero than $\lambda_2(\overline{\mathbf{W}})$). Thus, the iteration in (2) can be particularly slow if $\lambda_2(\overline{\mathbf{W}})$ is close to one and the vector $\mathbf{x}[0]$ has a nonzero projection onto the subspace spanned by the eigenvector corresponding to $\lambda_2(\overline{\mathbf{W}})$.

3. THE ACCELERATION ALGORITHM

In this section, we derive our novel algorithms and compare them with existing methods. We start by revisiting polynomial filters.

3.1 Polynomial Filtering

In our proposed method, we improve the convergence of (2) (in the mean sense) by using polynomial filters. The idea is similar to that proposed in [9], but the size of the filters that we use increase with the number of iterations. Later in this section we show that this is amenable to implementations with very low computational complexity and memory requirements. In addition, our method is optimal in a well defined sense (c.f. (7)) even if little information about $\mathbf{W}[i]$ in (2) is available.

In more detail, each agent k improves its local estimate of x_{av} by filtering $x_k[i]$ obtained with (1):

$$y_k[i] = \sum_{n=0}^i \gamma[i, n] x_k[n], \quad k \in \mathcal{N}, \quad (4)$$

where $\gamma[i, n] \in \mathbb{R}$ ($i \in \mathbb{N}$, $n = 0, \dots, i$) are *scalars to be designed* (common to all agents), and $y_k[i]$ is the improved estimate of x_{av} at time i in agent k . Stacking $y_1[i], \dots, y_N[i]$ in a vector $\mathbf{y}[i] := [y_1[i] \ \dots \ y_N[i]]^T$, we can rewrite (4) equivalently as

$$\mathbf{y}[i] = \sum_{n=0}^i \gamma[i, n] \mathbf{x}[n]. \quad (5)$$

Combining (3) and (5) and using Assumption 1, we can compute the mean value of $\mathbf{y}[i]$:

$$\begin{aligned} E[\mathbf{y}[i]] &= \sum_{n=0}^i \gamma[i, n] (\overline{\mathbf{W}})^n \mathbf{x}[0] \\ &= \mathbf{Q} \text{diag}(p_i(1), p_i(\lambda_2(\overline{\mathbf{W}})), \dots, p_i(\lambda_N(\overline{\mathbf{W}}))) \mathbf{Q}^T \mathbf{x}[0], \\ &= [(1/\sqrt{N})\mathbf{1} \ \mathbf{q}_2 \ \dots \ \mathbf{q}_N] \\ &\quad \cdot \text{diag}(p_i(1), p_i(\lambda_2(\overline{\mathbf{W}})), \dots, p_i(\lambda_N(\overline{\mathbf{W}}))) \\ &\quad \cdot [(1/\sqrt{N})\mathbf{1} \ \mathbf{q}_2 \ \dots \ \mathbf{q}_N]^T \mathbf{x}[0], \end{aligned} \quad (6)$$

where $p_i(x)$ is the polynomial $p_i(x) := \sum_{n=0}^i \gamma[i, n] x^n$ at time i . Now we need to choose a polynomial p_i that makes (6) a potentially better estimate of $x_{\text{av}}\mathbf{1}$ than (3).

3.2 The Synchronous Consensus Algorithm

By comparing (3) with (6), the slowest mode of convergence of $E[\mathbf{y}[i]]$ to $x_{\text{av}}\mathbf{1}$ is faster than that of $E[\mathbf{x}[i]]$ if the polynomials p_i satisfy the following properties: (see also [9], which, unlike the proposed method, use filters of short length.)

$$\text{P1) } p_i(1) = 1 \text{ and}$$

$$\text{P2) } \max_{j \in \{2, \dots, N\}} |p_i(\lambda_j(\overline{\mathbf{W}}))| < |\lambda_2(\overline{\mathbf{W}})|^i.$$

Therefore, at each time i , we conclude that we should find polynomials such that $p_i(1) = 1$ and that $|p_i(\lambda_j(\overline{\mathbf{W}}))|$ is as close to zero as possible for all $j \in \{2, \dots, N\}$ and all $i \in \mathbb{N}$. Unfortunately, finding an ideal polynomial having roots at $\lambda_2(\overline{\mathbf{W}}), \dots, \lambda_N(\overline{\mathbf{W}})$ (for $i \geq N-1$) would require global information about the network in every agent. To avoid this unrealistic requirement, we assume that the eigenvalues $\lambda_2(\overline{\mathbf{W}}), \dots, \lambda_N(\overline{\mathbf{W}})$ belong to the interval $[\alpha, \beta]$, but their exact values are unknown. (Assumption 1 guarantees $-1 < \alpha, \beta < 1$, and the bounds can be obtained from typical application scenarios; see Sect. 4.) With this assumption, a reasonable choice for p_i is the normalized polynomial $p_i(1) = 1$ of degree i least deviating from zero on the interval $[\alpha, \beta]$ (see also [11], [12, Sect. 10.1.5]). We can expect that such a polynomial would satisfy properties P1) and P2) above without knowledge of $\lambda_j(\overline{\mathbf{W}})$ ($j = 2, \dots, N$). More formally, at time i we use the polynomial:

$$p_i^* \in \arg \min_{p \in \mathcal{S}_i} \{ \max_{\alpha \leq x \leq \beta} |p(x)| \}, \quad (7)$$

where \mathcal{S}_i is the set of polynomials of degree i normalized to satisfy $p_i(1) = 1$. The polynomial in (7), which has been typically used to accelerate the convergence of iterative methods solving systems of linear equations, is unique and given by [11], [12, Sect. 10.1.5]:

$$p_i^*(x) = \frac{c_i \left(-1 + 2 \frac{x - \alpha}{\beta - \alpha} \right)}{c_i(\mu)},$$

where

$$\mu := 1 + 2 \frac{1 - \beta}{\beta - \alpha} \quad (8)$$

and c_i is the Chebyshev polynomial of degree i

$$c_i(x) = \begin{cases} \cos(i \cos^{-1} x), & |x| \leq 1, \ i \in \mathbb{N}, \\ \cosh(i \cosh^{-1} x), & |x| > 1, \ i \in \mathbb{N}. \end{cases}$$

Chebyshev polynomials can be generated with the recursion $c_{m+1}(x) = 2x c_m(x) - c_{m-1}(x)$ ($c_0(x) = 1$ and $c_1(x) = x$), so, similarly to the original Chebyshev acceleration algorithm [11], [12, Sect. 10.1.5], we can equivalently compute $E[\mathbf{y}[i]]$ (with the polynomial (7)) in the recursive form:

$$\begin{aligned} E[\mathbf{y}[i+1]] &= \omega_{i+1} [(1 - \kappa)\mathbf{I} + \kappa \overline{\mathbf{W}}] E[\mathbf{y}[i]] + (1 - \omega_{i+1}) E[\mathbf{y}[i-1]], \end{aligned} \quad (9)$$

where $E[\mathbf{y}[1]] = [(1 - \kappa)\mathbf{I} + \kappa \overline{\mathbf{W}}] \mathbf{x}[0]$, $\mathbf{y}[0] = \mathbf{x}[0]$, $\kappa := 2/(2 - \alpha - \beta)$, and

$$\omega_{i+1} = \frac{1}{1 - \frac{\omega_i}{4\mu^2}}, \quad i \geq 2, \quad \omega_1 = 1, \quad \omega_2 = \frac{2\mu^2}{2\mu^2 - 1}. \quad (10)$$

Unfortunately, unless $\mathbf{W}[i]$ is a constant matrix, the recursion in (9) cannot be implemented in a multiagent system because $\overline{\mathbf{W}}$ is

not available. Therefore, we replace expectations by sample values, and we obtain the following algorithm, which can be implemented in multiagent systems because the iteration in (1) (or, equivalently, (2)) can be readily implemented (using local computation of the resulting matrix-vector multiplications):

ALGORITHM 1. (*Synchronous Semi-Iterative Consensus Algorithm*)

$$\mathbf{z}[i+1] = \omega_{i+1} [(1-\kappa)\mathbf{I} + \kappa\mathbf{W}[i]] \mathbf{z}[i] + (1-\omega_{i+1})\mathbf{z}[i-1], \quad (11)$$

where $\mathbf{z}[1] = [(1-\kappa)\mathbf{I} - \kappa\mathbf{W}[0]]\mathbf{x}[0]$ and $\mathbf{z}[0] = \mathbf{x}[0]$.

The proposition below shows that some convergence properties of the original Chebyshev algorithm are retained, even though we replaced expectations by sample values.

PROPOSITION 1. (*Properties of the Synchronous Semi-Iterative Consensus Algorithm*)

Assume the conditions in Assumption 1. Then, the algorithm in (11) satisfies the following:

a) The algorithm is average preserving, i.e., $(1/N)\mathbf{1}^T \mathbf{z}[i] = (1/N)\mathbf{1}^T \mathbf{x}[0] = x_{\text{av}}$ for every $i \in \mathbb{N}$.

b) In the mean sense, the convergence of $\mathbf{z}[i]$ is identical to that of $\mathbf{y}[i]$ in (5) with $\gamma[i, n]$ chosen as the coefficients of the optimal polynomial in the sense of (7). In other words, $E[\mathbf{z}[i]] = E[\mathbf{y}[i]]$. Furthermore, if α and β are such that $-1 < \alpha \leq \min_{j \in \{2, \dots, N\}} \lambda_j(\overline{\mathbf{W}}) \leq \max_{j \in \{2, \dots, N\}} \lambda_j(\overline{\mathbf{W}}) \leq \beta < 1$, then

$$\|E[\mathbf{z}[i]] - x_{\text{av}}\mathbf{1}\| \leq \frac{\|\mathbf{x}[0]\|}{c_i(\mu)}, \quad (12)$$

which shows that $\lim_{i \rightarrow \infty} E[\mathbf{z}[i]] = x_{\text{av}}\mathbf{1}$ because $\lim_{i \rightarrow \infty} c_i(\mu) = \infty$.

PROOF. In the following, for notational convenience, we define: $\mathbf{M}[i] := [(1-\kappa)\mathbf{I} - \kappa\mathbf{W}[i]]$ and $\overline{\mathbf{M}} := [(1-\kappa)\mathbf{I} - \kappa\overline{\mathbf{W}}]$.

(a) By $\mathbf{1}^T \mathbf{W}[i] = \mathbf{1}^T$, we can check that

$$\mathbf{1}^T \mathbf{M}[i] = [(1-\kappa)\mathbf{1}^T \mathbf{I} - \kappa\mathbf{1}^T \mathbf{W}[i]] = \mathbf{1}^T,$$

and the result follows by induction. More precisely, note that $\mathbf{1}^T \mathbf{z}[0] = \mathbf{1}^T \mathbf{x}[0] = Nx_{\text{av}}$ and that $\mathbf{1}^T \mathbf{z}[1] = \mathbf{1}^T \mathbf{M}[0]\mathbf{x}[0] = N x_{\text{av}}$. Now, by assuming $\mathbf{1}^T \mathbf{z}[i] = N x_{\text{av}}$ and $\mathbf{1}^T \mathbf{z}[i-1] = N x_{\text{av}}$, we obtain

$$\begin{aligned} \mathbf{1}^T \mathbf{z}[i+1] &= \omega_{i+1} \mathbf{1}^T \mathbf{M}[i] \mathbf{z}[i] + (1-\omega_{i+1}) \mathbf{1}^T \mathbf{z}[i-1] \\ &= \omega_{i+1} N x_{\text{av}} + (1-\omega_{i+1}) N x_{\text{av}} = N x_{\text{av}}, \end{aligned}$$

which concludes the proof of (a).

(b) The proof can be informally shown as follows. From the i.i.d. assumption of the matrices $\mathbf{W}[i]$, we have that $\mathbf{z}[i]$ is independent of $\mathbf{W}[i]$, and thus $\mathbf{z}[i]$ is also independent of $\mathbf{M}[i]$. Now, apply the expectation operator in both sides of (11) to obtain

$$E[\mathbf{z}[i+1]] = \omega_{i+1} \overline{\mathbf{M}} E[\mathbf{z}[i]] + (1-\omega_{i+1}) E[\mathbf{z}[i-1]] \quad (13)$$

where $E[\mathbf{z}[1]] = \overline{\mathbf{M}}\mathbf{x}[0]$ and $E[\mathbf{z}[0]] = \mathbf{x}[0]$, and we conclude that $E[\mathbf{z}[i]] = E[\mathbf{y}[i]]$ for every $i \in \mathbb{N}$ (see (9)), and thus (13) is equivalent to (6) with $\mathbf{y}[i]$ replaced by $\mathbf{z}[i]$ and with p_i being the optimal polynomial in (7). Subtract $x_{\text{av}}\mathbf{1} = (1/N)\mathbf{1}\mathbf{1}^T \mathbf{x}[0]$ from both sides of (6) and use the facts that $|p_i^*(\lambda)| \leq 1/|c_i(\mu)|$ ($\alpha \leq \lambda \leq \beta$) and that $\|\mathbf{A}\mathbf{b}\|_2 \leq \|\mathbf{A}\|_2 \|\mathbf{b}\|_2$ for any matrix \mathbf{A} and vector \mathbf{b} of compatible sizes [12] to obtain (12). \square

Intuitively, Proposition 1 shows that our algorithm (with properly selected parameters α and β) is guaranteed to converge in the mean sense, and the convergence in the mean is typically faster

than that of the original scheme in (2). Unfortunately, unless the matrix $\mathbf{W}[i]$ is a constant matrix, the results in Proposition 1 are not enough to guarantee stability. In particular, Proposition 1 does not guarantee mean-square convergence, but this problem is also present in existing acceleration techniques that consider time-varying network matrices [8, 9]. However, in Sect. 4 we show that our method is robust in many practical scenarios. In addition, note that Proposition 1(a) holds even if the algorithm diverges (which could be the case when the parameter α is overestimated), so it can be useful to devise hybrid schemes with stronger convergence guarantees, but such methods are not investigated here.

3.3 The Asynchronous Consensus Algorithm

A potential limitation of Algorithm 1 is that agents should know the time instant i to compute ω_i . In some systems, such as those with agents communicating via network gossiping [7], knowing precisely the time index may not be possible. This fact renders Algorithm 1 impractical, and, to address this limitation, we propose an asynchronous semi-iterative consensus algorithm that is based on the following observation:

FACT 1. [11] [12, p. 517] (*On the convergence of ω_i*)

Let ω_i be as in (10) and $-1 < \alpha < \beta < 1$. Then, for $i > 1$, ω_i satisfies the following properties: i) $1 < \omega_i < 2$, ii) ω_i is strictly decreasing, and iii)

$$\lim_{i \rightarrow \infty} \omega_i = \frac{2}{1 + \sqrt{1 - 1/\mu^2}} =: \omega_\infty. \quad (14)$$

Since ω_i is convergent (and the asymptotic convergence is usually fast), we can try to approximate the output of Algorithm 1 by fixing ω_i to ω_∞ , the limit in (14). The resulting algorithm is formally presented below.

ALGORITHM 2. (*Asynchronous semi-iterative consensus algorithm*)

$$\mathbf{z}[i+1] = \omega_\infty [(1-\kappa)\mathbf{I} + \kappa\mathbf{W}[i]] \mathbf{z}[i] + (1-\omega_\infty)\mathbf{z}[i-1], \quad (15)$$

where $\mathbf{z}[1] = [(1-\kappa)\mathbf{I} - \kappa\mathbf{W}[0]]\mathbf{x}[0]$ and $\mathbf{z}[0] = \mathbf{x}[0]$.

It is not difficult to show that that Algorithm 2 is also average preserving and converges in the mean sense to $x_{\text{av}}\mathbf{1}$.

3.4 Relation with Existing Methods

We now compare the proposed algorithms against the original consensus algorithm with and without state-of-the-art acceleration methods. We start by rewriting (11) in the equivalent form:

$$\begin{aligned} z_k[i+1] &= \sum_{j \in \mathcal{N}_k} \omega_{i+1} (1-\kappa + \kappa[\mathbf{W}[i]_{kj}]) z_j[i] \\ &\quad + (1-\omega_{i+1}) z_k[i-1], \quad k \in \mathcal{N}, \quad (16) \end{aligned}$$

where $z_k[1] = \sum_{j \in \mathcal{N}_k} (1-\kappa + \kappa[\mathbf{W}[i]_{kj}) x_j[0]$ and $z_k[0] = x_k[0]$. (Note: Algorithm 2 is obtained by fixing ω_i in (16).)

From the above, we see that we need to keep two scalars in the memory of each agent, instead of one as in the original consensus algorithm in (1). In addition, in terms of local computation complexity per agent, Algorithm 1 is slightly more complex than the original consensus algorithm because (16) requires fewer additional sums and multiplications per iteration as compared to (1). However, the slightly higher computational complexity and memory requirements of the proposed method can be ignored because, in a real-world implementation with wireless links, agents spend most energy and time with the information exchange rather

than computation [13]. If agents implement either (1) or (16), they communicate with exactly the same neighbors and exchange the same amount of information per iteration. However, to reach the desired solution x_{av} within a prescribed precision, the iteration in (16) typically requires fewer iterations than the scheme in (1). As a result, the proposed methods can lead to great time and energy savings (because less information must be exchanged).

From the equivalence between $E[z[i]]$ and $E[y[i]]$ in (9) (or (6)), which is proved in Proposition 1(b), Algorithm 1 is applying a long polynomial filter that is optimal in a well defined sense and that uses all estimates $z[0], z[1], \dots$, even though its implementation only requires $z[i]$ and $z[i-1]$. This is in stark contrast with the implementation of the two algorithms in [9], both of which typically use short filters of fixed length and keep in the memory of each agent more than two samples of the sequence of estimates of x_{av} . One of the algorithms in [9] uses filters based on an intuitive approach that lacks an optimality criterion. The other approach in [9] uses filters optimal in a well defined sense, but it requires precise knowledge of the eigenvalues of $\overline{\mathbf{W}}$, which is an information not required by our approaches.

We can also relate our approaches with the two-register algorithms in [8, 10]. Assume that $\alpha = -\beta$ with $\beta > 0$, in which case $\kappa = 1$. Therefore, (16) reduces to:

$$z_k[i+1] = \sum_{j \in \mathcal{N}_k} \omega_{i+1} [\mathbf{W}[i]_{kj} z_j[i] + (1 - \omega_{i+1}) z_k[i-1]], \quad k \in \mathcal{N}, \quad (17)$$

where $z_k[0]$ and $z_k[1]$ are as in (16). As a result, the approaches in [8, Eq. (9)] and [10, Eq. (28)] (this last by also fixing the matrix $\mathbf{W}[i]$) are recovered if we fix ω_i to any number in the interval (1, 2). Fixing ω_i was also the approach used to derive Algorithm 2, which is an algorithm that gives strong arguments to use $\omega_i = \omega_\infty$ and not arbitrary values within the range (1, 2) (provided that the upper bound β is available). Note that we can also argue that fixing ω_∞ to an arbitrary value within the range (1, 2) is equivalent to making an arbitrary choice of β . More precisely, given $\omega_\infty \in (1, 2)$ and $\alpha = -\beta$ (which was used to derive (17)), we can use (14) to calculate the value of β that results in such a choice of ω_∞ . In the next section we show that the choice $\alpha = -\beta$ can be too pessimistic in many cases, and, as a result, the acceleration capabilities of the algorithm can be reduced.

A less strict reader could also argue that Algorithm 2 in its full generality is also equivalent to the algorithms in [8, Eq. (9)] and [10, Eq. (28)] with $\mathbf{W}[i]$ replaced by $(1 - \kappa)\mathbf{I} - \kappa\mathbf{W}[i]$. In such a case, these existing algorithms are accelerating the consensus algorithm $\mathbf{x}[i+1] = ((1 - \kappa)\mathbf{I} - \kappa\mathbf{W}[i])\mathbf{x}[i]$ and not the iteration in (2). Algorithm 2 shows how to choose κ (and ω_∞) when bounds on the eigenvalues are available.

4. EMPIRICAL EVALUATION

We now show the superiority of our proposed algorithms over existing methods, and we also evaluate the stability of the proposed algorithms in practical scenarios. In particular, as in [2], we place N agents uniformly at random in a square of unit area, and then we connect agents within distance $d = \sqrt{\log(N)/N}$ from each other (unless otherwise stated), which is a distance that guarantees that the resulting graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is connected with high probability [14]. We discard graphs not fully connected. In all simulations, each agent k initially reports values $x_k[0] = 50\sqrt{2} \cdot \|[\mathcal{X}_k \ \mathcal{Y}_k]^T\| + n_k$, where n_k is a sample of a Gaussian random variable with mean

zero and unit variance,² and \mathcal{X}_k and \mathcal{Y}_k are the Euclidean spatial coordinates of agent k in the unit grid. (Note that agents start with values strongly correlated with their positions as is common in the multiagent systems described earlier [2].)

We consider cases where agents exchange information not only with reliable communication links, but also with unreliable communication links because practical algorithms should be robust against link failures (a common occurrence in wireless links owing to the presence of jammers, obstacles, etc.). Therefore, for each scenario, the following network matrices $\mathbf{W}[i]$ are used:

- *(Reliable links.)* For simulations using reliable links, we set the network matrix to $\mathbf{W}[i] = \mathbf{I} - \epsilon\mathbf{L}$ ($i \in \mathbb{N}$), where \mathbf{L} is the Laplacian matrix of the graph \mathcal{G} and $\epsilon > 0$ is a properly selected scalar that guarantees that $\mathbf{W}[i]$ satisfies the conditions in Assumption 1.³ In particular, in this study we set $\epsilon = 0.05$ because it guaranteed the conditions in Assumption 1 in all our simulations. For a graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, the Laplacian matrix $\mathbf{L} \in \mathbb{R}^{N \times N}$ is given by $\mathbf{L} := \mathbf{D} - \mathbf{A}$, where $\mathbf{D} := \text{diag}(|\mathcal{N}_1| - 1, \dots, |\mathcal{N}_N| - 1)$ is the degree matrix ($|\cdot|$ denotes the cardinality of a set) and \mathbf{A} is the adjacency matrix [2, 4] $[\mathbf{A}]_{kj} = \begin{cases} 1, & \text{if } \{k, j\} \in \mathcal{E} \text{ and } k \neq j \\ 0, & \text{otherwise.} \end{cases}$

- *(Unreliable links.)* For this scenario, we use the model of unreliable links proposed in [5]. In more detail, we start with a connected graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ obtained as described above. At each time instant i , we copy all edges from \mathcal{E} to a new edge set $\mathcal{E}[i]$, and then we remove with probability p each edge $\{k, j\}$ ($k \neq j$) from $\mathcal{E}[i]$. The edge set $\mathcal{E}[i]$ defines a new graph $\mathcal{G}[i] = (\mathcal{N}, \mathcal{E}[i])$, and we use $\mathbf{W}[i] = \mathbf{I} - \epsilon\mathbf{L}[i]$, where $\mathbf{L}[i]$ is the Laplacian matrix associated with $\mathcal{G}[i] = (\mathcal{N}, \mathcal{E}[i])$. The physical interpretation of this model is that communication links, each of which corresponds to an edge in \mathcal{E} , can fail with probability p . As in the case with reliable links, we chose the value $\epsilon = 0.05$.

All acceleration methods in this section use the matrices $\mathbf{W}[i]$ described above. For convenience, we use the acronyms SSCA for the synchronous semi-iterative consensus algorithm and ASCA for the asynchronous semi-iterative consensus algorithm. The proposed acceleration schemes are compared with Laplacian-based methods without acceleration [2, 5] and with the following acceleration techniques:

- *The two-register eigenvalue shaping filter (ESF) in [10] (which is also the algorithm in [8] when the matrices $\mathbf{W}[i]$ are designed for consensus via network gossiping).* We showed in the discussion after (17) that this algorithm is equivalent to Algorithm 2 with $\alpha = -\beta$ and $\omega_\infty \in (1, 2)$. Our results are useful to help with the choice of ω_∞ when an upper bound on the second largest eigenvalue of the network matrix is available.

- *The optimal short-length polynomial filtering in [9] (algorithm denominated “polynomial” in the figures).* This algorithm uses more information than that available to the proposed schemes and the ESF acceleration method.

As in [8, 9], the performance metric of interest is the (absolute) squared error $\|\mathbf{o}[i] - x_{av}\mathbf{1}\|^2$, where $\mathbf{o}[i] \in \mathbb{R}^N$ is the output of a given consensus algorithm at time i (e.g., $\mathbf{o}[i] = \mathbf{z}[i]$ in the case of our proposed acceleration schemes or $\mathbf{o}[i] = \mathbf{x}[i]$ in the case of the original consensus algorithm in (2)). In simulations where the network matrix is deterministic, all algorithms are guaranteed

²The samples n_k are different in different runs of the simulation

³The matrices $\mathbf{W}[i]$ are fixed and deterministic in this scenario, so we can simply ignore the expectation operator in Assumption 1.

to converge if the eigenvalues of the network matrix (except for the eigenvalue 1) fall into the interval $[\alpha, \beta]$. Therefore, we plot the average squared error performance of the algorithms in such scenarios. However, unless otherwise stated, we show the (sample) 99th-percentile squared error performance when bounds on the eigenvalues are not exactly known. By plotting percentile curves, we identify algorithms that consistently provide good performance, and we also give less emphasis to rare situations where acceleration algorithms may not converge (see also the discussion after Proposition 1). Sample average and percentile curves are calculated from the results of 1,000 simulations, each of which use different initial conditions.

4.1 Networks with Reliable Links

Having described the basic setup of the simulations, we now study the performance of the algorithms in specific settings. We begin with an ideal scenario where the topology is fixed, links are reliable, and the minimum and second largest eigenvalues of the network matrix are precisely known. This scenario is useful to show the maximum acceleration gain that can be achieved with the algorithms because the minimum and second largest eigenvalue are simple to compute. The network under consideration is depicted in Fig. 1. For simplicity, denote $\mathbf{W} := \mathbf{W}[i]$ (because in this scenario the network matrices are fixed and deterministic), $\lambda_{\max} := \max_{j \in \{2, \dots, N\}} \lambda_j(\mathbf{W})$ ($\lambda_{\max} > 0$ in all our simulations), and $\lambda_{\min} = \min_{j \in \{2, \dots, N\}} \lambda_j(\mathbf{W})$. We use the following parameters for the proposed acceleration schemes: SSCA ($\alpha = \lambda_{\min}$, $\beta = \lambda_{\max}$) and ASCA ($\alpha = \lambda_{\min}$, $\beta = \lambda_{\max}$). The parameter c in [10, Eq. (28)] is set to $c = 1 - \omega_{\infty}$, where ω_{∞} is computed according to (14) with $\beta = -\alpha = \lambda_{\max}$ (see the discussion after (17) for the justification of this choice). Note that the ESF algorithm is basically the ASCA algorithm with $\beta = -\alpha$. We use filters of length eight for the method in [9] (this value is also used in [9]). Fig. 2 shows the performance of the algorithms.

As thoroughly discussed in [2], Laplacian-based algorithms have poor performance if the initial values reported by agents have a strong correlation with their positions, and this fact is indeed observed in Fig. 2. However, dramatic performance gains can be obtained by combining Laplacian-based methods with acceleration techniques. In particular, the SSCA and ASCA algorithms are able to provide in every agent values extremely close to x_{av} in very few iterations (as compared to the network size). The performance of the asynchronous algorithm ASCA closely follows that of its synchronous counterpart, the SSCA algorithm, which is optimal in a well defined sense. This result is not surprising because the asymptotic convergence of ω_i is fast. The performance advantage of the ASCA and SSCA algorithms over the ESF algorithm is explained by the fact that the former two algorithms use information about the lower bound on the eigenvalues of the network matrix. In contrast, the ESF algorithm, a particular case of the proposed method (see the discussion in Sect. 3.4), uses only information about the second largest eigenvalue (in magnitude), and the minimum eigenvalue is largely underestimated with the conservative lower bound $\alpha = -\beta$, which adversely affects the performance. The polynomial filtering scheme in [9], which requires precise knowledge of \mathbf{W} has performance comparable to the ESF algorithm. However, note that the scheme in [9] requires more information about the network matrix \mathbf{W} and has higher computation complexity than all other acceleration schemes. Its performance is inferior to that of the SSCA and ASCA algorithms because the proposed acceleration schemes are based on filters with increasing length.

We now study the stability of our proposed schemes when upper

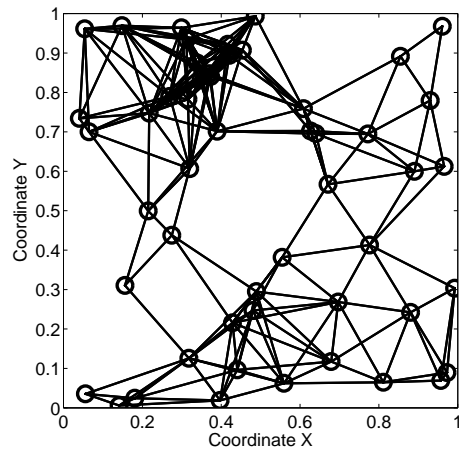


Figure 1: Network with 50 agents distributed uniformly at random in a square with unit area. Agents are represented by circles, and lines indicate communication links.

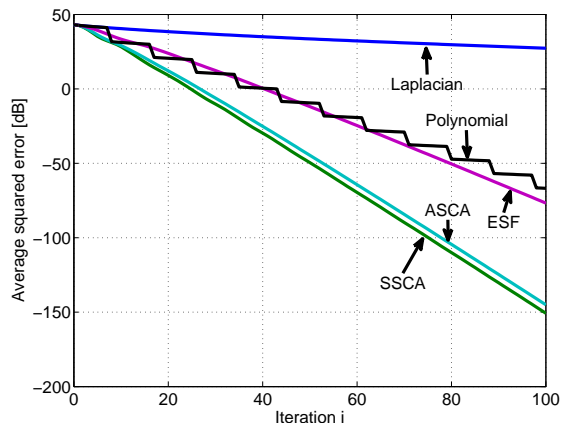


Figure 2: Transient performance of the algorithms. Every run of the simulation uses the topology in Fig 1.

and lower bounds of the eigenvalues of the network matrix are imprecisely estimated. For visual clarity, we use the network in Fig. 1 in all runs of the simulation, and we plot only results obtained with the original Laplacian-based algorithm (for reference) and the following versions of the SSCA algorithm:⁴ SSCA-under ($\beta = 0.9\lambda_{\max}$, $\alpha = \lambda_{\min}$) and SSCA-over ($\beta = \lambda_{\max}$, $\alpha = \lambda_{\min} + 0.05$). Note that SSCA-over underestimates the upper bound, whereas SSCA-under overestimates the lower bound. Fig. 3 shows the performance of the algorithms.

From Fig. 3 it is clear that, for the proposed algorithms, underestimating the upper bound is not so problematic as overestimating the lower bound. In contrast, neither convergence nor boundedness of $z[i]$ is guaranteed if α is overestimated because $|p_i^*(x)|$ grows fast outside $[\alpha, 1]$. This last fact explains the divergence of the SSCA-over algorithm.

In the simulations above, we have assumed exact knowledge of λ_{\max} and λ_{\min} , which is rarely the case in practice. Therefore,

⁴We omit the performance curves of the asynchronous algorithms because they are similar to those of the corresponding synchronous algorithms.

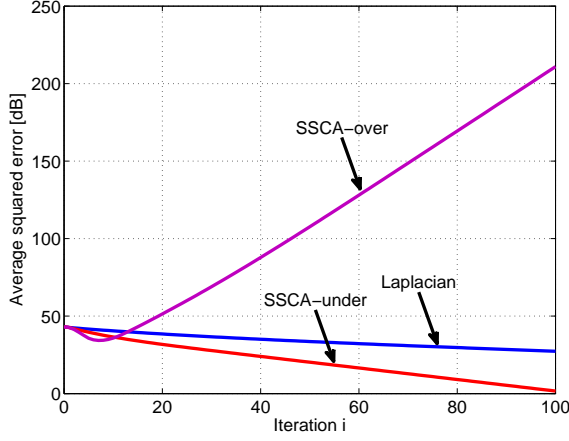


Figure 3: Stability of the algorithm with wrongly estimated bounds. Every run of the simulation uses the topology in Fig. 1.

to evaluate the algorithm in more practical settings, we consider a scenario where the topology changes at every run of the simulation. In such a case, α and β should be set to appropriate values based on likely bounds on the eigenvalues. Given that the proposed algorithms are robust against underestimated upper bounds, we can set β to a value expected to be greater than $|\lambda_2(\mathbf{W}[i])|$ with fairly high probability. However, we should take a conservative approach to choosing α because, as discussed above, overestimated values can render the proposed algorithms unstable. By simulating 100,000 different networks with the geometric approach described above, $|\lambda_2(\mathbf{W}[i])| \leq 0.994$ occurred in less than 1.32% of the simulations, so we choose $\beta = 0.994$ for both the SSCA and ASCA algorithms because we do not need to be overly conservative on the choice of β . As for the parameter α , we use $\alpha = -0.5$ because eigenvalues less than -0.5 have not been observed in our simulations. Therefore, we can expect that the proposed algorithms using $\alpha = -0.5$ converge with high probability. Fig. 4 shows the 99th-percentile squared error performance of the algorithms obtained by randomizing the network (and also the initial values reported by the agents) at each run of the simulation. In this figure, we once again set the parameter c in [10, Eq. (28)] to $c = 1 - \omega_\infty$, where ω_∞ was computed by using $0.994 = \beta = -\alpha$. We do not show the results of the polynomial filtering algorithm in [9] because, as in Fig. 2, its performance is worse than that obtained with other acceleration methods. In addition, it requires precise information about the network matrix in every run of the simulation, a very strong assumption in many multiagent systems.

With the settings in Fig. 4, the performance of Laplacian-based consensus methods is also poor, and all acceleration methods can greatly improve the convergence. The ASCA and SSCA algorithms were stable in all runs of our simulations, which is not surprising given the conservative choice of α . The ESF algorithm is basically the ASCA algorithm with an unduly underestimated parameter α , and this fact explains the worse performance of the ESF algorithm as compared to the SSCA and ASCA algorithms.

4.2 Networks with Unreliable Links

To study the stability of the acceleration algorithms with time-varying matrices $\mathbf{W}[i]$, we consider in Fig. 5 a scenario similar to that in Fig. 2, but with the difference that the communication links fail with probability $p = 0.2$ at each iteration (see the discussion in the beginning of this section). The parameters of all algorithms

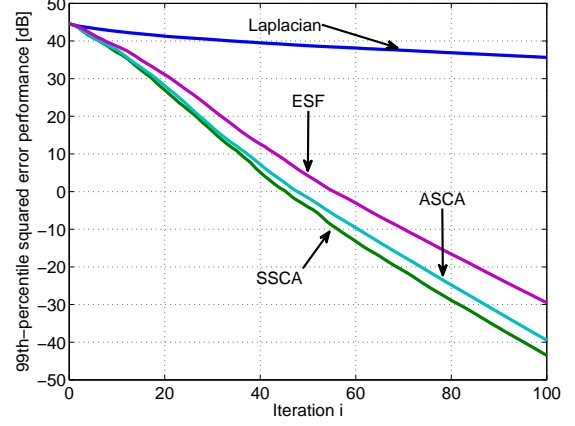


Figure 4: Transient performance of the algorithms. Network matrices $\mathbf{W}[i]$ are fixed and deterministic, but they change in every run of the simulation.

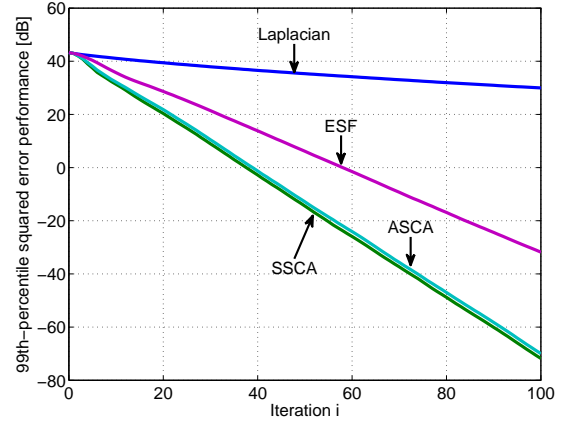


Figure 5: Transient performance of the algorithms. The network topology is the one in Fig. 1, but communication links fail with probability $p = 0.2$ at each iteration of the algorithms.

are the same as those in Fig. 2. We do not use exact bounds on the eigenvalues of $\overline{\mathbf{W}}$ to choose α and β because we want to illustrate a situation where the topology is supposed to be fixed and known, but the communication links are subject to failures that cannot be predicted (a common scenario in wireless networks). We omit once again the performance of the polynomial filtering approach in [9] because it did not converge in most runs of our simulations.

We can see in Fig. 5 that, with failing links, the proposed acceleration schemes (the ESF being a particular case) is acceptable because all agents are close to reach consensus on x_{av} in few iterations. In addition, the proposed algorithms converged in all runs of the simulation, which shows the good stability properties of our algorithms, even though Proposition 1 has only proved convergence in the mean sense. The relative performance of the algorithms is similar to that obtained in previous simulations, and the reason is the same as before.

In the last simulation, we study the impact of the network size on the convergence properties of the algorithms. In more detail, in Fig. 6 we show the (sample) median number of iterations that each

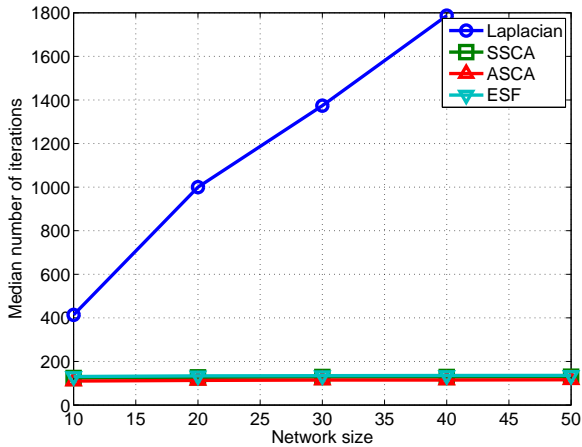


Figure 6: Median number of iterations required to reach the precision $\|o[i] - x[0]\|/\|x[0]\| \leq 0.001$ as a function of the network size. The network topology changes at each run of the simulation, and communication links can fail with probability $p = 0.2$ at each iteration of the algorithms.

algorithm requires to reach the precision $\|o[i] - x[0]\|/\|x[0]\| \leq 0.001$. In this figure, the network topology is randomized in every run of the simulation, and we decrease the connection range between agents to $d = 0.7\sqrt{\log(N)/N}$ to stress further the limitations of Laplacian-based methods and the gains obtained with acceleration algorithms. In addition, links fail with probability $p = 0.2$ at each iteration of the algorithms. If links are reliable, by keeping other conditions the same, the value 0.999 is a good estimate of the second largest eigenvalue of networks with sizes ranging from 10 to 50, so we use $\beta = 0.999$. More precisely, for networks of size 50, the second largest eigenvalue of the network matrix is greater than $\beta = 0.999$ with (empirical) probability less than 2% (with smaller networks, the probability is lower). For the minimum eigenvalue, eigenvalues less than -0.1 have not been observed in the simulations, so we use $\alpha = -0.1$. The parameters α and β have thus been adjusted to accommodate eigenvalues of (reliable) networks of different sizes and topologies. Note that the simulations in Fig. 6 use networks with unreliable links, and we did not try to estimate the eigenvalues of \overline{W} because the probability of failures cannot be usually predicted in real-world applications. The parameter c in [10, Eq. (28)] was once again set to $c = 1 - \omega_\infty$, where ω_∞ is given by (14) with $\beta = -\alpha$.

As proved in [2] and also observed in Fig. 6, Laplacian-based methods scale badly with the network size. However, all acceleration techniques are relatively insensitive to the network size, so they can be good alternatives to Laplacian-based methods in spatial computers. The compared acceleration schemes have similar performance because the precision, although fairly high, can be achieved in few iterations by all acceleration schemes (in previous simulations we can see that differences are usually more pronounced when we show 99th-percentile curves). Therefore, choosing parameters based on expected bounds on the eigenvalues of the network matrix (as proposed in this study) makes simple consensus algorithms practical in applications where approximate averages have to be computed with few iterations.

5. CONCLUSIONS

Laplacian-based methods for consensus have been identified as too slow to be practical in many multiagent applications, especially

those involving large-scale systems [2]. However, in this study we have demonstrated that such methods can still be useful in large systems if they are combined with acceleration techniques. In particular, the convergence speed of our two novel algorithms is fast and decreases gracefully with the network size in scenarios where the sole use of Laplacian-based methods are known to be impractical. Our first algorithm requires agents with synchronized clocks, and it is optimal in a well defined sense. The second algorithm is an asynchronous method that is able to provide performance very close to that of the optimal synchronous algorithm. Unlike existing acceleration methods, we have only assumed that rough bounds on the extremal eigenvalues of the network matrix are available (those bounds can be readily obtained by considering typical application scenarios).

6. REFERENCES

- [1] R. L. G. Cavalcante, A. Rogers, N. R. Jennings, and I. Yamada, "Distributed multiagent learning with a broadcast adaptive subgradient method," in *Proc. of the 9th Int. Conf. on Autonomous Agents and Multiagent Systems*, 2010, pp. 1039–1046.
- [2] N. Elhage and J. Beal, "Laplacian-based consensus on spatial computers," in *Proc. of the 9th Int. Conf. on Autonomous Agents and Multiagent Systems*, May 2010, pp. 907–914.
- [3] C.-H. Yu and R. Nagpal, "Sensing-based shape formation on modular multi-robot systems: A theoretical study," in *Proc. of the 7th Int. Conf. on Autonomous Agents and Multiagent Systems*, May 2008, pp. 71–78.
- [4] R. Olfati-Saber, J. A. Fax, and R. M. Murray, "Consensus and cooperation in networked multi-agent systems," *Proc. IEEE*, vol. 95, no. 1, pp. 215–233, Jan. 2007.
- [5] S. Kar and J. M. F. Moura, "Sensor networks with random links: Topology design for distributed consensus," *IEEE Trans. Signal Processing*, vol. 56, pp. 3315–3326, July 2008.
- [6] L. Xiao and S. Boyd, "Fast linear iterations for distributed averaging," *Systems and Control Letters*, vol. 53, pp. 65–78, Sept. 2004.
- [7] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, "Randomized gossip algorithms," *IEEE Trans. Inform. Theory*, vol. 52, no. 6, pp. 2508–2530, June 2006.
- [8] M. Cao, D. A. Spielman, and E. M. Yeh, "Accelerated gossip algorithms for distributed computation," in *Forty-Fourth Annual Allerton Conference*, Sept. 2006, pp. 952–959.
- [9] E. Kokiopoulou and P. Frossard, "Polynomial filtering for fast convergence in distributed consensus," *IEEE Transactions on Signal Processing*, vol. 57, no. 1, pp. 342–354, Jan. 2009.
- [10] D. Scherber and H. C. Papadopoulos, "Distributed computation of averages over ad hoc networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 4, pp. 776–787, Apr. 2005.
- [11] G. H. Golub and R. S. Varga, "Chebyshev semi-iterative methods, successive overrelaxation iterative methods, and second order Richardson iterative methods," *Numerische Mathematik*, no. 3, pp. 145–156, 1961.
- [12] G. H. Golub and C. F. V. Loan, *Matrix Computations*, 3rd ed. Baltimore, MD: The Johns Hopkins Univ. Press, 1996.
- [13] G. J. Pottie and W. J. Kaiser, "Wireless integrated network sensors," *Communications of the ACM*, vol. 43, no. 5, pp. 51–58, 2000.
- [14] P. Gupta and P. R. Kumar, "The capacity of wireless networks," *IEEE Transactions on Information Theory*, vol. 46, no. 2, pp. 388–404, Mar. 2000.

Game Theory I

Information Elicitation for Decision Making*

Yiling Chen
School of Engineering and
Applied Sciences
Harvard University
Cambridge, MA 02138 USA
yiling@eecs.harvard.edu

Ian A. Kash
Center for Research on
Computation and Society
Harvard University
Cambridge, MA 02138 USA
kash@seas.harvard.edu

ABSTRACT

Proper scoring rules, particularly when used as the basis for a prediction market, are powerful tools for eliciting and aggregating beliefs about events such as the likely outcome of an election or sporting event. Such scoring rules incentivize a single agent to reveal her true beliefs about the event. Othman and Sandholm [16] introduced the idea of a decision rule to examine these problems in contexts where the information being elicited is conditional on some decision alternatives. For example, “What is the probability having ten million viewers if we choose to air new television show X? What if we choose Y?” Since only one show can actually air in a slot, only the results under the chosen alternative can ever be observed. Othman and Sandholm developed proper scoring rules (and thus decision markets) for a single, deterministic decision rule: always select the the action with the greatest probability of success. In this work we significantly generalize their results, developing scoring rules for other deterministic decision rules, randomized decision rules, and situations where there may be more than two outcomes (e.g. less than a million viewers, more than one but less than ten, or more than ten million).

Categories and Subject Descriptors

J.4 [Computer Applications]: Social and Behavioral Sciences—*Economics*

General Terms

Economics, Theory

Keywords

Information Elicitation, Proper Scoring Rules, Decision Rules, Decision Markets, Mechanism Design, Market Design, Prediction Markets

*This paper is based upon work supported by NSF under Grant No. CCF-0953516. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the NSF.

Cite as: Information Elicitation for Decision Making, Yiling Chen and Ian A. Kash, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp.175-182

© 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

1. INTRODUCTION

Eliciting information about uncertain events is crucial for informed decision making. Information is often possessed by individual agents. Truthfully eliciting such information, resolving conflicting beliefs, and aggregating the dispersed information are key problems for achieving collective intelligence in multi-agent systems. Proper scoring rules have been proposed to incentivize an expert to honestly report her probability assessment for an uncertain event [2, 7, 21, 22, 18, 23, 3, 6, 11]. Prediction markets, betting intermediaries where participants wager on the outcome of some event of interest, can output equilibrium market prices representing the consensus probability assessment based on the pooled information of market participants [1, 24, 17].

However, most work, with the exception of Othman and Sandholm [16] and Dimitrov and Sami [4], separates the information elicitation problem from the decision problem. Information elicitation and aggregation are studied without considering how the resultant information will be used. Participants of mechanisms are assumed to only care about their rewards within the mechanism. Incentives for the isolated elicitation problem may not motivate desirable behavior of participants when the elicited information is used for decision making, because participants may have vested interests in some decision outcome and may misreport their information hoping to achieve the decision outcome.

In this paper, we study the information elicitation problem situated in a decision making process. We consider a setting where a decision maker seeks to elicit information about the consequences of various actions that he could take and choose an action based on the elicited information. For example, a company needs to choose among three marketing campaigns and wants to elicit the probability that each campaign will result in a sales goal being reached. This setting was first investigated by Othman and Sandholm [16] when the decision maker uses a deterministic decision rule and there are two possible outcomes given any action. They derived a proper scoring rule for a particular decision rule. *Our main contribution is a theorem that significantly generalizes their result by characterizing all (strictly) proper scoring rules for all decision rules.*

The rest of the paper is organized as follows. In Section 2, we cover the relevant background and related work on the use of scoring rules for information elicitation. Then in Section 3, we provide a formal model of the problem of eliciting information from a single expert for the purpose of making a decision. We state and prove our main result, a characterization theorem for (strictly) proper scoring rules in the

decision making setting, in Section 4. Using this characterization, we provide a sufficient condition for the existence of a strictly proper scoring rule and a sufficient condition for when no such rule exists in Section 5. For situations where strictly proper scoring rules are not possible, we provide a derivation of “quasi-strictly proper” scoring rules in Section 6. We discuss issues that arise when eliciting information from multiple experts and some potential solutions in Section 7. We conclude in Section 8.

2. BACKGROUND AND RELATED WORK

Proper scoring rules have been designed to incentivize a risk-neutral expert to truthfully report her probability assessment for an uncertain event [2, 7, 21, 22, 18, 23, 14, 13, 20, 15]. Let v be a discrete random variable that has m mutually exclusive and exhaustive outcomes, $\mathcal{O} = \{o_1, \dots, o_m\}$. A scoring rule assigns score $s_o(\vec{p})$ to an expert who reports a probability assessment \vec{p} when outcome o happens. A scoring rule is *regular* if $s_o(\vec{p})$ is real valued for all o , except possibly $s_o(\vec{p}) = -\infty$ if $p_o = 0$. A regular scoring rule is (strictly) *proper* if a risk-neutral expert (strictly) maximizes her expected score by reporting truthfully. That is, \vec{p} is an optimal solution to $\max_{\vec{r} \in \Delta(\mathcal{O})} \sum_{o=1}^m p_o s_o(\vec{r})$ for any proper scoring rule s , and is the unique optimal solution if s is strictly proper. $\Delta(\mathcal{O})$ is the probability simplex over \mathcal{O} . For example, logarithmic scoring rule $s_o(\vec{p}) = a_o + b \log p_o$ and quadratic scoring rule $s_o(\vec{p}) = a_o + b(2p_o - \sum_i p_i^2)$, where $b > 0$ and a_o are arbitrary parameters, are strictly proper scoring rules.

Proper scoring rules are closely related to convex functions. In fact, the following characterization theorem of Gneiting and Raftery [6], which is credited to McCarthy [12] and Savage [18], gives the precise relationship between convex functions and proper scoring rules. In the theorem, \cdot denotes the vector inner product.

THEOREM 2.1 (GNEITING AND RAFTERY THEOREM 2).
A regular scoring rule is (strictly) proper if and only if

$$s_o(\vec{p}) = G(\vec{p}) - G'(\vec{p}) \cdot \vec{p} + G'_o(\vec{p}),$$

where $G : \Delta(\mathcal{O}) \rightarrow \mathbb{R}$ is a (strictly) convex function and $G'(\vec{p})$ is a subgradient of G at the point \vec{p} and $G'_o(\vec{p})$ is the o -th element of $G'(\vec{p})$.

Theorem 2.1 indicates that a regular scoring rule is (strictly) proper if and only if its expected score function $G(\vec{p}) = \sum_o p_o s_o(\vec{p})$ is (strictly) convex on $\Delta(\mathcal{O})$, and the vector with elements $s_o(\vec{p})$ is a subgradient of G at the point \vec{p} .

Hanson [9, 10] shows how a proper scoring rule designed to elicit information from a single expert can be turned into a mechanism for prediction markets, which aggregate the information of multiple experts. Such a mechanism is called a *market scoring rule*, and is essentially a shared proper scoring rule. Hanson [8] also promotes the idea of *decision markets*. A decision market is a prediction market for conditional events. A decision maker who needs to decide among some actions can operate a conditional prediction market for each action. The conditional market elicits information on outcomes of some event of interest conditioned on the corresponding action being taken (e.g. probability that stock price of a company increases conditioned on A is hired as the CEO). The decision maker can decide on what action to take based on the elicited conditional probability distributions. Our work focuses on the incentive problem of eliciting

conditional information from a single expert using a scoring rule, but we also discuss the implications for using a market scoring rule for decision markets. Furthermore, while Hanson proposes the idea of a decision market, he does not provide any analysis or techniques showing how one could be implemented to correctly encourage participants to reveal their information. As we discuss in Section 7, this is a difficult problem.

The closest work to ours is that of Othman and Sandholm [16]. They pair a scoring rule for eliciting conditional probability distributions over two outcomes with a deterministic decision rule. This differs from the standard information elicitation problem using proper scoring rules, because only one action will be taken and used to determine the score of an expert, but the selected action depends on the reported conditional probability distributions. Othman and Sandholm show that for deterministic decision rules, to have a “quasi-strictly proper” scoring rule it is necessary that the decision rule only change its decision when probabilities are equal. A natural version of this is the MAX rule: decide on the action with the highest reported probability for the more desirable outcome. They construct a quasi-strictly proper scoring rule for MAX and then extend their results to decision markets. However, they show that it is impossible to achieve properness in decision markets using the MAX decision rule. As an open problem, they pose the question of how proper scoring rules can be derived for randomized decision rules. Our main theorem answers this question with a characterization of all (strictly) proper scoring rules for all decision rules. Thus, we extend their results to deterministic rules other than MAX, randomized decision rules, and situations with more than two outcomes.

Three other papers have considered the problem of information elicitation in other settings where the outcome is not independent of the predictions of experts. Shi, Conitzer, and Guo [19] examine settings where participants in a prediction market may also have an ability to influence the outcome. For example, participants in a market to predict terrorist attacks may be able to carry out acts of terrorism and employees of a company participating in a prediction market about when a new product will launch may have the ability to delay the launch. They show how to derive scoring rules that do not incentivize the participants to take these “undesirable” actions. However, unlike our work, the information elicited is not explicitly used for decision making. Dimitrov and Sami [4] examine incentive problems when there are two prediction markets for different but related events. This might cause a trader to report sub-optimally in one market to mislead a trader in another market. The first trader can also participate in the second market and profit from correcting the second trader. This is a situation where the payoff from the first market does not depend on the decision the second trader makes; instead, the first trader profits directly from the decision. Our work considers the opposite case: experts do not care what decision is made, except that the outcome (and thus their payoffs) depends on it. Gerding et al. [5] consider a model where experts need to be incentivized to make costly observations of the quality of service providers. They consider a number of approaches, including scoring rules, and all face tradeoffs between encouraging experts to invest effort and getting accurate reports. One of their approaches, basing scores on peer predictions, could potentially be helpful with resolving the issues with decision

markets discussed in Section 7.

3. OUR MODEL

Before we introduce our model, we note some notational conventions we use throughout the paper. We typically denote a matrix with a capital letter and an entry of a matrix P as $P_{i,o}$. We denote a vector as \vec{p} , except that when considering a row of a matrix as a vector we denote it P_i . We use the Frobenius inner product, $P : Q = \sum_{i,o} P_{i,o} Q_{i,o}$, for matrices.

Our model is essentially that of Othman and Sandholm [16], but adapted to allow randomized decision rules and more than two outcomes.

A decision maker needs to choose an action from a set $\mathcal{A} = \{1, \dots, n\}$. Each action may affect the probability of achieving each possible outcome from a set $\mathcal{O} = \{o_1, \dots, o_m\}$. Othman and Sandholm [16] considered the case of two outcomes, in which case we use \top to denote the more desirable outcome and \perp to denote the less.

The decision maker asks an expert to report a set of conditional probability distributions, denoted by a $n \times m$ matrix P , where $P_{i,o}$ is the probability of outcome o conditional on the decision maker taking action i . We use P_i to denote the i -th row of P , that is, the probability distribution over \mathcal{O} conditional on action i being taken. $P_i \in \Delta(\mathcal{O})$ for all actions i , where $\Delta(\mathcal{O})$ is the probability simplex over \mathcal{O} . We use \mathcal{P} to denote the space of P . In general, not every decision need potentially lead to each outcome. For example, we could model a decision maker that cares about which decision is made by having a disjoint set outcomes for each decision.

Based on the expert's report, the decision maker makes a decision using a decision rule $D : \mathcal{P} \rightarrow \Delta(\mathcal{A})$. $D_i(P)$ is the probability the decision maker assigns to action i given report P . In the special case of a deterministic decision rule, $D : \mathcal{P} \rightarrow \mathcal{A}$. The decision rule D is known by the expert.

To encourage the expert to make an accurate prediction, the decision maker rewards her using a scoring rule $S : \mathcal{A} \times \mathcal{O} \times \mathcal{P} \rightarrow \mathbb{R} \cup \{-\infty\}$. For notational convenience, we use $S_{i,o}(P)$ to represent $S(i, o, P)$, the score for the report P when action i is taken and outcome o happens. Note that we allow the expert's reward to depend on the decision made, a feature not necessary for the deterministic decision rules considered in Othman and Sandholm's model. We assume the expert is risk neutral and only cares about her reward according to the scoring rule. Specifically, she does not care what decision is made, other than to the extent that it affects her expected score.

We now define regular, proper, strictly proper, and quasi-strictly proper scoring rules for a decision rule.

DEFINITION 3.1. *A scoring rule S is regular for decision rule D if $S_{i,o}(P) \in \mathbb{R}$ unless $P_{i,o} = 0$.*

The definition is analogous to that of regular scoring rules in Section 2. An expert may get a score of $-\infty$ only if an event occurred to which she assigned probability 0. We consider only regular scoring rules for a decision rule in this paper because if this condition is not met an expert can get $-\infty$ in expectation, making the scoring rule unappealing.

Let $V(P, Q)$ denote the expected score of an expert who believes that the true conditional probabilities are P but reports the probabilities Q , i.e. $V(P, Q) = \sum_{i,o} D_i(Q) P_{i,o} S_{i,o}(Q)$.

We define (strictly) proper scoring rules for a decision rule as follows, which is a direct generalization of (strictly) proper scoring rules in Section 2.

DEFINITION 3.2. *A regular scoring rule S is proper for a decision rule D if*

$$V(P, P) \geq V(P, Q)$$

for all P and all $Q \neq P$. It is strictly proper for the decision rule if the inequality is strict.

Othman and Sandholm showed that no deterministic decision rule has a strictly proper scoring rule, but showed one that satisfies a slightly weaker condition. Intuitively, the decision maker does not care if the expert is not strictly incentivized to tell the truth about the probabilities for actions he does not take as long as he learns the true conditional probabilities for the action he takes. We formally define the notion for randomized decision rules below.

DEFINITION 3.3. *A regular scoring rule S is quasi-strictly proper for a decision rule D if it is proper (i.e.*

$$V(P, P) \geq V(P, Q)$$

for all P and all $Q \neq P$), and

$$V(P, P) > V(P, Q)$$

for all P and Q such that $P_k \neq Q_k$ for some $k \in \sigma_Q$, where $\sigma_Q = \{i | D_i(Q) > 0\}$ is the support of $D(Q)$ ¹.

A quasi-strictly proper scoring rule for a decision rule ensures that an expert is strictly incentivized to truthfully report her conditional probability distributions for actions that will be taken with positive probabilities by the decision maker, although she may lie about her conditional probability distributions for actions that won't be taken without changing her expected score.

4. CHARACTERIZING PROPER SCORING RULES FOR DECISION RULES

In this section, we state and prove our main theorem, a characterization of all regular (strictly) proper scoring rules for arbitrary (randomized) decision rules. We show that any scoring rule of a particular form is (strictly) proper for the corresponding decision rule and that every regular (strictly) proper scoring rule for a decision rule is of this form. This form, similar to the one used by Gneiting and Raftery in Theorem 2.1, relies on the (strict) convexity of a function G , which can be thought of as the expected truthful score function $V(P, P) = \sum_{i,o} D_i P_{i,o} S_{i,o}(P)$. Our theorem can be interpreted as saying that a scoring rule is (strictly) proper for D if and only if $G(P) = V(P, P)$ is (strictly) convex and satisfies some additional conditions. G need not be differentiable in general (for example with a deterministic decision rule), so rather than using the gradient of G , the theorem uses the notion of a subgradient. At a point where G is differentiable, the gradient is the unique subgradient.

¹Othman and Sandholm give a different definition of quasi-strict properness, which does not account for the possibility that a decision rule may be effectively "tied." For example the scoring rule they give for the MAX decision rule violates their definition when $P_{1,\top} = 0.5$, $P_{2,\top} = 0.5$, $Q_{1,\top} = 0.4$, and $Q_{2,\top} = 0.5$, but satisfies our definition.

The resulting theorem is quite powerful. For an arbitrary decision rule and scoring rule it provides a simple test to determine whether the scoring rule is proper for the decision rule. For an arbitrary decision rule, it gives a method of constructing proper scoring rules. Additionally, generalizations of many of Othman and Sandholm's results [16] characterizing properties of proper scoring rules for deterministic decision rules to situations where the decision rule does not have full support (for example that there are no strictly proper scoring rules and that all proper scoring rules satisfy an independence of irrelevant alternatives condition), are simple corollaries of our theorem.

THEOREM 4.1. *A regular scoring rule is (strictly) proper for a decision rule D if and only if*

$$S_{i,o}(P) = \begin{cases} G(P) - G'(P) : P + \frac{G'_{i,o}(P)}{D_i(P)} & D_i(P) > 0 \\ \Pi_{i,o}(P) & D_i(P) = 0 \end{cases} \quad (1)$$

where $G : \mathcal{P} \rightarrow \mathbb{R} \cup \{-\infty\}$ is a (strictly) convex function, $G'(P)$ is a subgradient of G at the point P with $G'_{i,o}(P) = 0$ when $D_i(P) = 0$, and $\Pi_{i,o} : \mathcal{P} \rightarrow \mathbb{R} \cup \{-\infty\}$ is an arbitrary function that can take a value of $-\infty$ only when $P_{i,o} = 0$.

PROOF. Consider a regular scoring rule S satisfying (1). We first show that it must be (strictly) proper. Let $\sigma_P = \{i \mid D_i(P) > 0\}$. We have,

$$\begin{aligned} V(P, P) &= \sum_{i,o} D_i(P) P_{i,o} S_{i,o}(P) \\ &= \sum_{i \in \sigma_P, o} D_i(P) P_{i,o} \left(G(P) - G'(P) : P + \frac{G'_{i,o}(P)}{D_i(P)} \right) \\ &= G(P) - G'(P) : P + \sum_{i \in \sigma_P, o} G'_{i,o}(P) P_{i,o} \\ &= G(P) - G'(P) : P + G'(P) : P = G(P). \end{aligned}$$

The fourth equality relies on the condition that $G'_{i,o}(P) = 0$ when $D_i(P) = 0$. Because G is convex and G' is a subgradient, for $Q \neq P$

$$\begin{aligned} V(P, Q) &= \sum_{i,o} D_i(Q) P_{i,o} S_{i,o}(Q) \\ &= \sum_{i \in \sigma_Q, o} D_i(Q) P_{i,o} \left(G(Q) - G'(Q) : Q + \frac{G'_{i,o}(Q)}{D_i(Q)} \right) \\ &= G(Q) + (P - Q) : G'(Q) \leq G(P) = V(P, P). \end{aligned}$$

This gives us that S is a proper scoring rule for D . The inequality is strict if G is strictly convex, in which case S is strictly proper for D .

Now consider a regular proper scoring rule S for D . We will show that it must be of the form of (1). Define $G(P) = V(P, P) = \sup_Q V(P, Q)$ and $G'_{i,o}(P) = D_i(P) S_{i,o}(P)$. Each $V(P, Q)$ is a convex function of P . G is a point-wise supremum of a set of convex functions and hence is convex itself. If $D_i(P) = 0$, $G'_{i,o}(P) = 0$ by definition. For $Q \neq P$, we have

$$\begin{aligned} G(P) + (Q - P) : G'(P) &= \sum_{i,o} D_i(P) P_{i,o} S_{i,o}(P) + \sum_{i,o} (Q_{i,o} - P_{i,o}) D_i(P) S_{i,o}(P) \\ &= \sum_{i,o} D_i(P) Q_{i,o} S_{i,o}(P) = V(Q, P) \leq V(Q, Q) = G(Q). \end{aligned}$$

The inequality is due to the properness of S . It is strict if S is strictly proper. Thus, G' is a subgradient of the convex function G , and G is strictly convex if S is strictly proper. Finally, for any P and j such that $D_j(P) > 0$,

$$\begin{aligned} G(P) - G'(P) : P + \frac{G'_{j,o'}(P)}{D_j(P)} &= \sum_{i,o} D_i(P) P_{i,o} S_{i,o}(P) - \sum_{i,o} D_i(P) S_{i,o}(P) P_{i,o} + \frac{G'_{j,o'}(P)}{D_j(P)} \\ &= \frac{G'_{j,o'}(P)}{D_j(P)} = \frac{D_j(P) S_{j,o'}(P)}{D_j(P)} = S_{j,o'}(P). \end{aligned}$$

So, S is of the proper form. \square

In Theorem 4.1, $\Pi_{i,o}$ allows arbitrary scores to be assigned when $D_i(P) = 0$, subject to the constraint that the scoring rule is regular. As the decision rule never takes action i , the score that would be assigned if it did is essentially arbitrary and does not affect the expected score of the expert. The theorem has another condition with no parallel in Theorem 2.1 by Gneiting and Raftery: the requirement that $G'_{i,o}(P) = 0$ if $D_i(P) = 0$. This can be read as requiring the expert's expected score to be independent of her reports about the probabilities for actions that will not be taken. For proper scoring rules for a decision rule D , this condition is satisfied by the trivial convex function $G(P) = 0$, which pays the expert nothing no matter what she reports, so she is weakly indifferent to truthful reporting. However, this condition may not be satisfied by any strictly convex function, resulting in a non-existence of strictly proper scoring rules for D , an issue to which we return in Section 5.

We conclude this section with a number of examples of proper scoring rules that can be derived using Theorem 4.1.

- For the two-outcome case, taking $G(P) = \max_i P_{i,\top}$ for the MAX decision rule gives the proper scoring rule derived by Othman and Sandholm [16].
- More generally, with more than two outcomes the decision maker may have some utility $u(o)$ for each outcome and want to use the deterministic decision rule that selects the action i that maximizes expected utility $U_i(P) = \sum_o u(o) P_{i,o}$. In this case, he can use $G(P) = \max_i U_i(P)^2$, which gives the proper scoring rule $S_{i,o} = 2U_i(P)u(o) - U_i(P)^2$. Note that this rule is not strictly proper, but we will see in Section 6 that it is quasi-strictly proper.
- For the two outcome case with randomized decision rule $D_i(P) = P_{i,\top} / \sum_j P_{j,\top}$, taking $G(P) = \sum_i P_{i,\top}^2$ gives us the strictly proper scoring rule $S_{i,\top} = \sum_j 2P_{j,\top} - P_{j,\top}^2$ and $S_{i,\perp} = -\sum_j P_{j,\top}^2$, which is reminiscent of the quadratic scoring rule.

5. STRICT PROPERNESS

In addition to characterizing all proper scoring rules for a particular decision rule, Theorem 4.1 characterizes the strictly proper scoring rules as well. However, as Othman and Sandholm [16] observed for the case of deterministic rules, some decision rules may not have any strictly proper scoring rules. More generally, we would like to know whether, given a decision rule D , there exists a strictly convex G satisfying the requirements of Theorem 4.1, and thus a strictly

proper scoring rule. In this section, we give sufficient conditions for both the existence and non-existence of strictly proper scoring rules.

For a strictly proper scoring rule to exist, we need to find a strictly convex function G that satisfies the condition from Theorem 4.1 that $G'_{i,o}(P) = 0$ whenever $D_i(P) = 0$. When $D(P)$ always has full support (i.e. $D_i(P)$ is never 0) this is trivially satisfied by any strictly convex function. This gives us a sufficient condition for a decision rule to have a strictly proper scoring rule.

COROLLARY 5.1. *If a decision rule D always has full support ($D_i(P) > 0$ for all i and P) then it has a strictly proper scoring rule.*

PROOF. Any strictly convex function G , for example $G(P) = \sum_{i,o} P_{i,o}^2$, satisfies the requirements of Theorem 4.1 and thus yields a strictly proper scoring rule. \square

On the other hand, Othman and Sandholm show that no deterministic decision rule has a strictly proper scoring rule. The following corollary establishes a larger class of decision rules for which this is the case, namely those for which the probability distribution over actions chosen by the decision rule does not have full support and there is a case where the probabilities outside the support can be changed without changing the probability distribution over actions.

COROLLARY 5.2. *If there exist $P \neq Q$ such that*

1. $D(P) = D(Q)$,
2. $\sigma_P \subset \mathcal{A}$, and
3. $P_{i,o} = Q_{i,o}$ for all $i \in \sigma_P$ and all o ,

then D does not have a strictly proper scoring rule.

PROOF. Consider such a P and Q and a proper scoring rule S . By Theorem 4.1, $G'_{i,o}(P) = G'_{i,o}(Q) = 0$ for all $i \notin \sigma_P$, so $G(P) = G(Q)$ and

$$\begin{aligned} V(P, Q) &= \sum_{i,o} P_{i,o} D_i(Q) S_{i,o}(Q) \\ &= \sum_{i \in \sigma_Q, o} P_{i,o} D_i(Q) (G(Q) + G'(Q) : Q + G'_{i,o}(Q)) \\ &= \sum_{i \in \sigma_P, o} P_{i,o} D_i(P) (G(P) + G'(P) : P + G'_{i,o}(P)) \\ &= V(P, P). \end{aligned}$$

Thus S is not strictly proper. \square

While Corollary 5.2 shows that a subset of decision rules that do not have full support do not have a strictly proper scoring rule, the following open problem remains.

OPEN PROBLEM 1. *Characterize when decision rules that do not have full support and also do not satisfy the additional conditions of Corollary 5.2 have a strictly proper scoring rule.*

6. QUASI-STRICT PROPERNESS

We saw in Section 5 that, while we can always construct a proper scoring rule, many decision rules do not have any strictly proper scoring rules. The mere existence of a proper scoring rule is unsatisfying; the scoring rule that gives the expert a score of 0 no matter what the decision and outcome

is proper for every decision rule but gives the expert no particular incentive to reveal her beliefs. Strictly proper scoring rules fix this problem by ensuring that truthful reporting is uniquely optimal. While not quite as satisfying, a quasi-strictly proper scoring rule provides the weaker promise that, no matter what optimal report the expert makes, she reported her true beliefs over the outcome space *for the actions the decision maker might take*. In this section, we give a derivation of quasi-strictly proper scoring rules for a class of decision rules.

To build intuition about how quasi-strictly proper scoring rules can be derived, we first examine a set of sufficient conditions for a scoring rule to be quasi-strictly proper for the MAX decision rule.

LEMMA 6.1 (OTHMAN AND SANDHOLM [16]). *Let f and g be functions such that*

1. f and g are twice differentiable on $(0, 1)$,
2. $h(p) = pf(p) + (1-p)g(p)$ is strictly increasing on $[0, 1]$,
3. $pf'(p) + (1-p)g'(p) = 0$ for all $p \in [0, 1]$, and
4. $pf''(p) + (1-p)g''(p) < 0$ for all $p \in [0, 1]$.²

Then $S_{i,\top}(P) = f(P_{i,\top})$ and $S_{i,\perp}(P) = g(P_{i,\top})$ is quasi-strictly proper for the MAX decision rule.

A subset of these conditions also suffices to prove that a function h used in the construction is strictly convex.

LEMMA 6.2. *Let f and g be functions such that*

1. f and g are twice differentiable on $(0, 1)$,
2. $pf'(p) + (1-p)g'(p) = 0$ for all $p \in [0, 1]$,
3. $pf''(p) + (1-p)g''(p) < 0$ for all $p \in [0, 1]$.

Then $h(p) = pf(p) + (1-p)g(p)$ is strictly convex on $[0, 1]$.

PROOF. $h'(p) = f(p) - g(p) + pf'(p) + (1-p)g'(p)$ and $h''(p) = 2(f'(p) - g'(p)) + pf''(p) + (1-p)g''(p)$. Because $pf'(p) + (1-p)g'(p) = 0$, we have $h'(p) = f(p) - g(p)$ and $h''(p) = f'(p) - g'(p)$. Combining our two equations for h'' gives $f'(p) - g'(p) + pf''(p) + (1-p)g''(p) = 0$, or $f'(p) > g'(p)$. Thus $h''(p) > 0$ and h is strictly convex. \square

This is not a coincidence; we now show how such strictly convex functions can be used to construct quasi-strictly proper scoring rules for a large class of decision rules. In the simple case of deterministic decision rules, the members of this class share the feature that the desirability of each action can be computed as a strictly convex function of the conditional probabilities reported for that action and the decision rule simply takes the maximum of these desirabilities. For example, the MAX decision rule for two outcomes can be expressed as $D(P) \in \operatorname{argmax}_i P_{i,\top}^2$, where $h(P_i) = P_{i,\top}^2$ is a strictly convex function of P_i . More generally the decision rule may randomize over several actions and may a priori exclude some actions or combinations of actions from consideration. Thus, our construction proceeds by selecting a subset of the power set of actions, associating a strictly convex function with each, and showing that every corresponding decision rule has a quasi-strictly proper scoring rule.

²Othman and Sandholm do not explicitly state this condition, but it is implicit from the proof of their Theorem 7.

LEMMA 6.3. Let $\beta \subseteq 2^{\mathcal{A}} - \{\emptyset\}$, and for each $b \in \beta$, $G^b : \Delta(\mathcal{O})^{|\beta|} \rightarrow \mathbb{R}$ be a strictly convex function, $D(P)$ have support $\text{argmax}_{b \in \beta} G^b(P_b)$ for all P , and P_b be the submatrix of P consisting of those rows whose action is in b . Then, the scoring rule from Theorem 4.1 with $G(P) = \max_{b \in \beta} G^b(P_b)$ is quasi-strictly proper for D .

PROOF. Let P and Q be given and let b be the support of $D(Q)$ (the decisions made with positive probability). Then by Theorem 4.1 and the strict convexity of G^b ,

$$\begin{aligned} V(P, Q) &= G(Q) + (P - Q) : G'(Q) \\ &= G^b(Q_b) + (P_b - Q_b) : (G^c)'(Q_b) \\ &\leq G^b(P_b) \leq \max_{c \in \beta} G^c(P_c) = V(P, P). \end{aligned}$$

If $P_b \neq Q_b$ the first inequality is strict. If $b \notin \text{argmax}_{c \in \beta} G^c(P_c)$ the second is strict. Thus the scoring rule is quasi-strictly proper for D . \square

In the statement of Lemma 6.3, β is the set of possible supports the decision rule considers. For each such support b , G^b is a strictly convex function that determines how “good” that support is given the probabilities ($G^b(P_b)$). Our construction applies to any decision rule that always has a support that is “best” according to the various G^b ($\text{argmax}_{b \in \beta} G^b(P_b)$). There are many such rules, as this condition restricts only the support, not the actual decision probabilities. Each has a different quasi-strictly proper scoring rule, but they can all be derived from the same convex function $G(P) = \max_{b \in \beta} G^b(P_b)$.

Lemma 6.3 allows us to derive quasi-strictly proper scoring rules for deterministic decision rules with two outcomes using $\beta = \mathcal{A}$ and $G^b = h$. For example, we can derive quasi-strictly proper scoring rules for MAX (e.g. $h(P_i) = P_{i,\top}^2$, mentioned previously, gives Othman and Sandholm’s rule [16]), MIN (e.g. $h(P_i) = P_{i,\perp}^2$), and even strange rules like “probability farthest from 0.5” ($h(P_i) = (P_{i,\top} - 0.5)^2$). We can take $\beta \subset \mathcal{A}$ to allow for decisions rules that only allow certain actions (e.g. “choose whichever of actions 1 and 3 is more likely to succeed”). With more than two outcomes it allows rules like the expected utility maximization rule from Section 4. We can also apply this construction to the randomized case. For example, we saw in Section 4 a construction of a scoring rule that is strictly proper for the decision rule $D_i(P) = P_{i,\top} / \sum_j P_{j,\top}$. Lemma 6.3 tells us that a version of this rule that disregards some actions and uses an appropriately modified scoring rule is quasi-strictly proper. In particular, if $\alpha \subset \mathcal{A}$ is the set of actions considered then Lemma 6.3 can be applied with $\beta = \{\alpha\}$ and $G^\alpha(P) = \sum_{i \in \alpha} P_{i,\top}^2$.

The proof of Lemma 6.3 actually proves something stronger than quasi-strict properness. In particular, it shows that, unless the support of $D(Q)$ is a maximizer of $\max_{c \in \beta} G^c(P_c)$, $V(P, P) > V(P, Q)$. Thus, not only does the expert have a strict incentive to report the true probabilities for the actions the decision maker ends up randomizing over, she also has a strict incentive to ensure this set is one that the decision rule considers “optimal.”

One interesting observation about these scoring rules in the deterministic case is that they can all be viewed as strictly proper scoring rules when outcomes are exogenous. For example taking $h(P_i) = P_{i,\top}^2$ and $D(P) = i$ gives the scoring rule $s_{\top}(P_i) = 2P_{i,\top} - P_{i,\top}^2$ and $s_{\perp}(P_i) = -P_{i,\top}^2$,

which is a variant of the well known quadratic scoring rule (which is strictly proper).

In fact, we can show that this is generally true for quasi-strictly proper scoring rules derived according to Lemma 6.3 with a deterministic decision rule. When D is a deterministic decision rule, its support given P must be a singleton action. Hence, β in Lemma 6.3 equals \mathcal{A} . For each element of β (i.e. each action i), we set $G^i(P_i) = h(P_i)$, where h is strictly convex. Thus, the decision rule will take action k where $k \in \text{argmax}_{i \in \mathcal{A}} h(P_i)$. We assume that the decision rule breaks ties arbitrarily when there are more than one actions that have the same highest value of $h(P_i)$. We have $G(P) = \max_{i \in \mathcal{A}} h(P_i) = h(P_k)$ and can derive a quasi-strictly proper scoring rule $S_{i,o}(P)$ according to expression (1) in Theorem 4.1. Clearly, for the chosen action k , $S_{k,o}(P)$ only depends on P_k . We would like to consider whether $S_{k,o}(P)$ can be viewed as a strictly proper scoring rule of P_k assuming that action k is always chosen no matter how P_k changes. Let $\vec{q} = \text{argmin}_{\vec{p}} h(\vec{p})$. \vec{q} is unique because h is strictly convex. We construct a scoring rule $s_o(\vec{p}) = S_{k,o}(Q^{\vec{p}})$ where $Q^{\vec{p}}$ is a probability matrix where $Q_k^{\vec{p}} = \vec{p}$ and $Q_j^{\vec{p}} = \vec{q}$ for all $j \neq k$.

COROLLARY 6.1. $s_o(\vec{p})$ constructed above is strictly proper.

PROOF. For all \vec{p} , $D(Q^{\vec{p}}) = k$, so $s_o(\vec{p}) = S_{k,o}(Q^{\vec{p}}) = h(\vec{p}) - h'(\vec{p}) \cdot \vec{p} + h'_o(\vec{p})$. By Theorem 2.1 and the strict convexity of h , s is strictly proper. \square

Thus, for deterministic decision rules, the quasi-strictly proper scoring rules derived according to Lemma 6.3 are strictly proper given a chosen action. As we will see in Section 7.1, this is a potentially useful property if one of these rules is used as a basis for a decision market.

7. DISCUSSION: DECISION MARKETS

Scoring rules are useful in their own right as a tool to elicit information from a single expert, but collectively a group of experts may provide better information. In this section, we discuss some challenges and observations on using decision markets to elicit information from multiple experts.

For a standard market scoring rule, using a strictly proper scoring rule s , the market maintains a probability distribution over outcomes \vec{p} . At any time, a trader can change this to \vec{q} , and in doing so accepts the following bet: if outcome o occurs then the market pays her $s_o(\vec{q}) - s_o(\vec{p})$ (which may be negative). A trader who only participates once maximizes her expected payoff by changing the market probability to match her true beliefs. We can use the same approach for decision markets. A decision market maintains a market probability matrix P . Any trader can change this to Q , accepting the bet that if action i is taken and outcome o occurs then she receives $S_{i,o}(Q) - S_{i,o}(P)$. At the close of the market, there is some final probability matrix F , and the decision is made according to $D(F)$.

However, as Othman and Sandholm [16] observed, traders’ incentives are not as perfectly aligned in a decision market, even if S is a proper scoring rule for D . A trader’s payoff relying on $D(F)$ points to two key issues. First, in order to determine her expected utility for a report, a trader needs to know what F will be, which is not determined until the market closes. One way to resolve this issue is to follow Othman and Sandholm and consider the last trader in the market, whose report is F (or equivalently assume that

traders are myopic and all assume they are the last trader). Second, in a standard market scoring rule, a trader's expected payment to the market institution given beliefs \vec{q} is $\vec{q} \cdot s(\vec{p})$, which is independent of her report. Thus, she chooses her report \vec{r} to maximize $\vec{q} \cdot s(\vec{r})$, and strict properness of s is sufficient. However, a myopic trader who reports R in a decision market makes an expected payment of $\sum_{i,o} D_i(R) Q_{i,o} S_{i,o}(P)$ given beliefs Q . This is *not* independent of R . Thus, although properness of S means that Q maximizes $\sum_{i,o} D_i(R) Q_{i,o} S_{i,o}(R)$, unlike the simple market scoring rule case, it does not follow that Q maximizes $\sum_{i,o} D_i(R) Q_{i,o} (S_{i,o}(R) - S_{i,o}(P))$.

Othman and Sandholm give the following example for the two-outcome, two-action case under the MAX decision rule with scoring rule $S_{i,\top}(P) = 2P_{i,\top} - P_{i,\top}^2$, $S_{i,\perp}(P) = -P_{i,\top}^2$ and show that all scoring rules for MAX have similar manipulations. Suppose a trader believes the true probabilities are $(Q_{1,\top}, Q_{2,\top}) = (0.8, 0.75)$, but the current market probabilities are $(0.8, 0.3)$. If the trader reports her true belief, her net expected payment is 0, but if she reports $(0.8, 0.81)$ her expected payment is 0.15. In essence, she only gets paid for correcting the value of $P_{2,\top}$ if she convinces the decision maker to choose decision 2. To make matters worse, any later trader with similar beliefs is weakly indifferent to correcting the market, so the market may get stuck at these wrong probabilities. Furthermore, this manipulation is "safe;" if action 1 is chosen in the end the trader's payment is 0 and if action 2 is chosen her expected payment is positive.

Clearly this is not a desirable outcome. Othman and Sandholm propose to address this problem choosing a scoring rule that minimizes, but does not eliminate, the incentive for a trader to perform such a manipulation. In the remainder of this section we consider several other approaches.

7.1 Faith in Markets

Suppose that, rather than being myopic, a trader believes the market will "get it right" in the end. That is, if her beliefs are Q , she believes that, regardless of her report F will eventually equal Q . Then she believes that the portion of her payment to the market institution based on the current market probabilities is $\sum_{i,o} D_i(Q) Q_{i,o} S_{i,o}(P)$, which is independent of her report. Thus she wants to optimize $\sum_{i,o} D_i(Q) Q_{i,o} S_{i,o}(R)$ by selecting R . In this case, $S_{i,o}(R)$ need not to be proper for D . In fact, we can replace $S_{i,o}(R)$ with any standard proper scoring rule $s(R_i)$ and traders are incentivized to report their true beliefs. Thus, if traders believe in the market, the decision maker can simply use a standard proper scoring rule!

Of course, as Othman and Sandholm's example shows, traders may have good reason to believe that the market will not get it right. In particular, traders near the close of the market may have an incentive to distort the probabilities. Luckily, for deterministic decision rules, Corollary 6.1 shows that the quasi-strictly proper scoring rules we derive are in fact proper scoring rules in this sense! Thus by using such a rule we can simultaneously provide myopic traders an incentive for truthful reporting in many, though not all, situations and provide traders with faith in the market an incentive for truthful reporting.

7.2 Differing Beliefs

We saw that one potential way around Othman and Sand-

holm's example is if some traders have a different belief about the final market prediction F . Another possibility is some traders have different beliefs about the true probability matrix. For example, consider a trader arriving with beliefs $(0.79, 0.74)$ and market probabilities $(0.8, 0.81)$. Depending on her beliefs about F , the trader has an incentive to change at least one of the probabilities to match her beliefs, so the market will no longer be "stuck" at $(0.8, 0.81)$.

7.3 Randomized Decision Rules

The negative example involves a deterministic decision rule. Potentially, randomized rules could have better incentive properties. However, simply adding randomness is not a panacea, as the following example shows.

Suppose $\mathcal{A} = \{1, 2\}$ and $\mathcal{O} = \{\top, \perp\}$. In section 4, we saw that the scoring rule $S_{i,\top} = \sum_j 2P_{j,\top} - P_{j,\top}^2$ and $S_{i,\perp} = \sum_j -P_{j,\top}^2$ is strictly proper for the decision rule $D_i(P) = P_{i,\top} / (P_{1,\top} + P_{2,\top})$. Suppose the current market probabilities are $(0.8, 0.7)$ and a myopic trader arrives with the belief $(0.8, 0.7)$. Ideally, we would like her to not make a prediction as the current market probabilities match her belief. However, it turns out to be optimal for her to report $(0.75, 0.75)$. More generally, we have the following lemma.

LEMMA 7.1. *Suppose $\mathcal{A} = \{1, 2\}$, $\mathcal{O} = \{\top, \perp\}$, $D_i(P) = P_{i,\top} / (P_{1,\top} + P_{2,\top})$, and $S_{i,\top} = \sum_j 2P_{j,\top} - P_{j,\top}^2$ and $S_{i,\perp} = \sum_j -P_{j,\top}^2$. Suppose the current market probabilities are P and a myopic trader arrives with belief P . The trader's optimal report is $Q_{i,\top} = (P_{1,\top} + P_{2,\top})/2$. Furthermore, suppose the current market probabilities are Q when the trader arrives. Then her optimal report is still Q .*

The proof is straightforward calculus and is therefore omitted. In many ways this example is worse than Othman and Sandholm's. Even if the market reaches the correct probabilities, the next trader to arrive will change them to values that give the decision maker no useful information. Furthermore, these uninformative values are stable. While this feature makes this particular randomized decision rule and scoring rule pair a poor choice for use in a decision market, the more general question is open.

OPEN PROBLEM 2. *Is there a randomized decision rule and corresponding scoring rule with good incentive properties for decision markets?*

7.4 Increasing Market Maker Loss

Another option is to consider a more drastic change to the design of the market. Othman and Sandholm's example shows that a decision market can get "stuck" with a prediction like $(0.8, 0.81)$ that no rational agent has an incentive to fix, because of the form of the expected payment of a myopic trader: $\sum_{i,o} D_i(R) Q_{i,o} (S_{i,o}(R) - S_{i,o}(P))$. Suppose we gave each trader only the side of the bet based on her prediction (i.e. $\sum_{i,o} D_i(R) Q_{i,o} (S_{i,o}(R))$ assuming she is myopic). Then if S is proper she would have an incentive to report her true probability rather than leaving the market at the current probability.

This approach loses the shared nature of market scoring rules and may create a large loss for the market maker. In particular, using a market scoring rule, if the initial prediction is P^0 and traders update this as P^1, \dots, P^f , the market maker's total payments to traders when action i is chosen

and outcome o occurs are $S_{i,o}(P^1) - S_{i,o}(P^0)$, $S_{i,o}(P^2) - S_{i,o}(P^1)$, \dots , $S_{i,o}(P^f) - S_{i,o}(P^{f-1})$, for a total of $S_{i,o}(P^f) - S_{i,o}(P^0)$. If traders' payments are changed to be based only on their own predictions, this property disappears.

While paying each trader based solely on her own prediction can be expensive, if this is done occasionally the loss may be acceptable. For example, once per hour or once per day the market maker could select a random trader to whom to make such an offer. The loss of the market maker is linear in the number of such offers made.

8. CONCLUSION

We examined the problem of information elicitation for decision making. One agent, a decision maker, wants to choose a distribution over a set of actions based on the probability distribution over outcomes for each action. Another agent, an expert, has a belief about these probabilities that the decision maker wants to elicit. Such elicitation is done through scoring rules. Othman and Sandholm [16] studied this problem for deterministic decision rules, with many of their results focusing on the MAX decision rule in particular. Our main result significantly generalized their results by providing a complete characterization of (strictly) proper scoring rules for arbitrary decision rules.

This characterization allowed us to give a sufficient condition for a decision rule to have a strictly proper scoring rule and a sufficient condition for no strictly proper scoring rule to exist. As these sufficient conditions do not cover all decision rules, an open problem remains. We also showed how our characterization allows us to derive quasi-strictly proper scoring rules in a number of cases where strictly proper scoring rules do not exist.

Finally, we discussed how the elicitation problem becomes more complicated when there are multiple experts, an observation also made by Othman and Sandholm [16]. A natural approach is to use a proper scoring rule for a decision rule to make a decision market, in the same way proper scoring rules are used to make prediction markets. However, this introduces two main problems. First, since only one decision is made in the end, an agent trading in the market has to base her decisions on beliefs about what the final market probabilities would be, a strategic problem with no parallel in prediction markets. Second, since no individual trader controls the final decision, scoring rules that encourage truthful revelation when they do have control no longer have the safe effect. We examined several ways this problem might be tackled in practice.

9. REFERENCES

- [1] J. E. Berg, R. Forsythe, F. D. Nelson, and T. A. Rietz. Results from a dozen years of election futures markets research. In C. A. Plott and V. Smith, editors, *Handbook of Experimental Economic Results*. 2001.
- [2] G. W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.
- [3] J. Cervera and J. Munoz. Proper scoring rules for fractiles. *Bayesian Statistics*, 5:513–519, 1996.
- [4] S. Dimitrov and R. Sami. Composition of markets with conflicting incentives. In *EC '10: Proceedings of the 11th ACM conference on Electronic commerce*, pages 53–62, 2010.
- [5] E. Gerding, K. Larson, and N. Jennings. Eliciting expert advice in service-oriented computing. In *Proceedings of the 11th International Workshop on Agent-Mediated Electronic Commerce*, pages 29–42, 2009.
- [6] T. Gneiting and A. E. Raftery. Strictly proper scoring rules, prediction, and estimation. *J. of the American Statistical Association*, 102(477):359–378, March 2007.
- [7] I. J. Good. Rational decisions. *Journal of the Royal Statistical Society, Series B (Methodological)*, 14(1):107–114, 1952.
- [8] R. Hanson. Decision markets. *IEEE Intelligent Systems*, 14(3):16–19, 1999.
- [9] R. D. Hanson. Combinatorial information market design. *Information Systems Frontiers*, 5(1):107–119, 2003.
- [10] R. D. Hanson. Logarithmic market scoring rules for modular combinatorial information aggregation. *Journal of Prediction Markets*, 1(1):1–15, 2007.
- [11] N. Lambert, D. M. Pennock, and Y. Shoham. Eliciting properties of probability distributions. In *EC '08: Proceedings of the 9th ACM conference on Electronic commerce*, pages 129–138, 2008.
- [12] J. McCarthy. Measures of the value of information. *PNAS: Proceedings of the National Academy of Sciences of the United States of America*, 42(9):654–655, 1956.
- [13] A. H. Murphy and R. L. Winkler. Probability forecasting in meteorology. *Journal of the American Statistical Association*, 79(387):489–500, 1984.
- [14] R. G. Nelson and D. A. Bessler. Subjective probabilities and scoring rules: Experimental evidence. *Journal of Agricultural Economics*, 71(2):363–369, 1989.
- [15] F. M. O'Carroll. Subjective probabilities and short-term economic forecast: An empirical investigation. *Applied Statistics*, 26(3):269–278, 1977.
- [16] A. Othman and T. Sandholm. Decision rules and decision markets. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 625–632, 2010.
- [17] D. M. Pennock and R. Sami. Computational aspects of prediction markets. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*. Cambridge University Press, 2007.
- [18] L. J. Savage. Elicitation of personal probabilities and expectations. *Journal of the American Statistical Association*, 66(336):783–801, 1971.
- [19] P. Shi, V. Conitzer, and M. Guo. Prediction mechanisms that do not incentivize undesirable actions. In *WINE '09: Proceedings of the 5th International Workshop on Internet and Network Economics*, pages 89–100, 2009.
- [20] D. J. Spiegelhalter. Probabilistic prediction in patient management and clinical trials. *Statistics in Medicine*, 5(5):421–433, 1986.
- [21] R. L. Winkler. The quantification of judgment: Some methodological suggestions. *Journal of the American Statistical Association*, 62(320):1105–1120, 1967.
- [22] R. L. Winkler. Scoring rules and the evaluation of probability assessors. *Journal of the American Statistical Association*, 64(327):1073–1078, 1969.
- [23] R. L. Winkler, J. Mu'oz, J. Cervera, J. Bernardo, G. Blattenberger, J. Kadane, D. Lindley, A. Murphy, R. Oliver, and D. Ršos-Insua. Scoring rules and the evaluation of probabilities. *TEST: An Official Journal of the Spanish Society of Statistics and Operations Research*, 5(1):1–60, June 1996.
- [24] J. Wolfers and E. Zitzewitz. Prediction markets. *Journal of Economic Perspective*, 18(2):107–126, 2004.

Stable partitions in additively separable hedonic games

Haris Aziz Felix Brandt Hans Georg Seedig

Department of Informatics
Technische Universität München
85748 Garching bei München, Germany
{aziz,brandtf,seedig}@in.tum.de

ABSTRACT

An important aspect in systems of multiple autonomous agents is the exploitation of synergies via coalition formation. In this paper, we solve various open problems concerning the computational complexity of stable partitions in additively separable hedonic games. First, we propose a polynomial-time algorithm to compute a contractually individually stable partition. This contrasts with previous results such as the NP-hardness of computing individually stable or Nash stable partitions. Secondly, we prove that checking whether the core or the strict core exists is NP-hard in the strong sense even if the preferences of the players are symmetric. Finally, it is shown that verifying whether a partition consisting of the grand coalition is contractual strict core stable or Pareto optimal is coNP-complete.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent Systems; J.4 [Computer Applications]: Social and Behavioral Sciences—*Economics*

General Terms

Theory, Economics

Keywords

Game theory (cooperative and non-cooperative); teamwork, coalition formation, coordination; incentives for cooperation

1. INTRODUCTION

Ever since the publication of von Neumann and Morgenstern’s *Theory of Games and Economic Behavior* in 1944, coalitions have played a central role within game theory. The crucial questions in coalitional game theory are which coalitions can be expected to form and how the members of coalitions should divide the proceeds of their cooperation. Traditionally the focus has been on the latter issue, which led to the formulation and analysis of concepts such as the core, the Shapley value, or the bargaining set. Which coalitions are likely to form is commonly assumed to be settled exogenously, either by explicitly specifying the coalition

Cite as: Stable partitions in additively separable hedonic games, Haris Aziz, Felix Brandt and Hans Georg Seedig, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 183-190.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

structure, a partition of the players in disjoint coalitions, or, implicitly, by assuming that larger coalitions can invariably guarantee better outcomes to its members than smaller ones and that, as a consequence, the grand coalition of all players will eventually form. The two questions, however, are clearly interdependent: the individual players’ payoffs depend on the coalitions that form just as much as the formation of coalitions depends on how the payoffs are distributed.

Coalition formation games, as introduced by Drèze and Greenberg [12], provide a simple but versatile formal model that allows one to focus on coalition formation. In many situations it is natural to assume that a player’s appreciation of a coalition structure only depends on the coalition he is a member of and not on how the remaining players are grouped. Initiated by Banerjee et al. [4] and Bogomolnaia and Jackson [6], much of the work on coalition formation now concentrates on these so-called *hedonic games*. Hedonic games are relevant in modeling many settings such as formation of groups, clubs and societies [6], and also online social networking [13]. The main focus in hedonic games has been on notions of stability for coalition structures such as *Nash stability*, *individual stability*, *contractual individual stability*, or *core stability* and characterizing conditions under which the set of stable partitions is guaranteed to be non-empty (see, e.g., [6, 8]). Sung and Dimitrov [21] presented a taxonomy of stability concepts which includes the *contractual strict core*, the most general stability concept that is guaranteed to exist. A well-studied special case of hedonic games are two-sided matching games in which only coalitions of size two are admissible [18]. We refer to Hajduková [16] for a critical overview of hedonic games.

Hedonic games have recently been examined from an algorithmic perspective (see, e.g., [3, 11]). Cechlárová [9] surveyed the algorithmic problems related to stable partitions in hedonic games in various representations. Ballester [3] showed that for hedonic games represented by *individually rational list of coalitions*, the complexity of checking whether core stable, Nash stable, or individual stable partitions exist is NP-complete. He also proved that every hedonic game admits a contractually individually stable partition. Coalition formation games have also received attention in the artificial intelligence community where the focus has generally been on computing optimal partitions for general coalition formation games without any combinatorial structure [19]. Elkind and Wooldridge [13] proposed a fully-expressive model to represent hedonic games which encapsulates well-known representations such as *individually rational list of coalitions* and *additive separability*.

Additively separable hedonic games (ASHGs) constitute a particularly natural and succinctly representable class of hedonic games. Each player in an ASHG has a value for any other player and the value of a coalition to a particular player is simply the sum of the values he assigns to the members of his coalition. Additive separability satisfies a number of desirable axiomatic properties [5] and ASHG are the non-transferable utility generalization of *graph games* studied by Deng and Papadimitriou [10]. Olsen [17] showed that checking whether a nontrivial Nash stable partition exists in an ASHG is NP-complete if preferences are nonnegative and symmetric. This result was improved by Sung and Dimitrov [22] who showed that checking whether a core stable, strict core stable, Nash stable, or individually stable partition exists in a general ASHG is NP-hard.

Dimitrov et al. [11] obtained positive algorithmic results for subclasses of ASHG in which each player merely divides other players into friends and enemies. Branzei and Larson [7] examined the tradeoff between stability and social welfare in ASHG. Recently, Gairing and Savani [14] showed that computing partitions that satisfy some variants of individual-based stability is PLS-complete, even for very restricted preferences. In another paper, Aziz et al. [2] studied the complexity of computing and verifying optimal partitions in ASHG.

In this paper, we settle the complexity of key problems regarding stable partitions of ASHG. We present a polynomial-time algorithm to compute a contractually individually stable partition. This is the first positive algorithmic result (with respect to one of the standard stability concepts put forward by Bogomolnaia and Jackson [6]) for general ASHG with no restrictions on the preferences. We strengthen recent results of Sung and Dimitrov [22] and prove that checking whether the core or the strict core exists is NP-hard, even if the preferences of the players are symmetric. Finally, it is shown that verifying whether a partition is in the contractual strict core (CSC) is coNP-complete, even if the partition under question consists of the grand coalition. This is the first computational hardness result concerning CSC stability in hedonic games of any representation. The proof can be used to show that verifying whether the partition consisting of the grand coalition is Pareto optimal is coNP-complete, thereby answering a question mentioned by Aziz et al. [2]. Our computational hardness results imply computational hardness of the equivalent questions for *hedonic coalition nets* [13].

2. PRELIMINARIES

In this section, we provide the terminology and notation required for our results.

A *hedonic coalition formation game* is a pair (N, \mathcal{P}) where N is a set of players and \mathcal{P} 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 the set $\mathcal{N}_i = \{S \subseteq N \mid i \in S\}$. The statement $S \succ_i T$ denotes that i strictly prefers S over T whereas $S \sim_i T$ means 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 of π that includes player i .

We consider utility-based models rather than purely ordinal models. In *additively separable preferences*, a player i gets 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

$\sum_{j \in S} v_i(j)$. A game (N, \mathcal{P}) is *additively separable* if for each player $i \in N$, there is a utility function $v_i : N \rightarrow \mathbb{R}$ such that $v_i(i) = 0$ and for coalitions $S, T \in \mathcal{N}_i$, $S \succsim_i T$ if and only if $\sum_{j \in S} v_i(j) \geq \sum_{j \in T} v_i(j)$. We will denote the utility of player i in partition π by $u_\pi(i)$.

A preference profile 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 $i, j \in N$. For any player i , let $F(i, A) = \{j \in A \mid v_i(j) > 0\}$ be the set of *friends of player i within A* .

We now define important stability concepts used in the context of coalition formation games.

- A partition is *Nash stable (NS)* if no player can benefit by moving from his coalition S to another (possibly empty) coalition T .
- A partition is *individually stable (IS)* if no player can benefit by moving from his coalition S to another existing (possibly empty) coalition T while not making the members of T worse off.
- A partition is *contractually individually stable (CIS)* if no player can benefit by moving from his coalition S to another existing (possibly empty) coalition T while making neither the members of S nor the members of T worse off.
- We say that a coalition $S \subseteq N$ *strongly 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)*.
- We say that 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 in the *contractual strict core (CSC)* if any weakly blocking coalition S makes at least one player $j \in N \setminus S$ worse off when breaking off.

The inclusion relationships between stability concepts depicted in Figure 1 follow from the definitions of the concepts. We will also consider *Pareto optimality*. A partition π of N is *Pareto optimal* if there exists no partition π' of N such that for all $i \in N$, $\pi'(i) \succsim_i \pi(i)$ and there exists at least one player $j \in N$ such that $\pi'(j) \succ_j \pi(j)$. We say that a partition π satisfies *individual rationality* if each player does as well as by being alone, i.e., for all $i \in N$, $\pi(i) \succsim_i \{i\}$.

Throughout the paper, we assume familiarity with basic concepts of computational complexity (see, e.g., [1]).

3. CONTRACTUAL INDIVIDUAL STABILITY

It is known that computing or even checking the existence of Nash stable or individually stable partitions in an ASHG is NP-hard. On the other hand, a potential function argument can be used to show that at least one CIS partition exists for every hedonic game [3]. The potential function argument does not imply that a CIS partition can be computed in polynomial time. There are many cases in hedonic games, where a solution is guaranteed to exist but *computing* it is not feasible. For example, Bogomolnaia and Jackson [6]

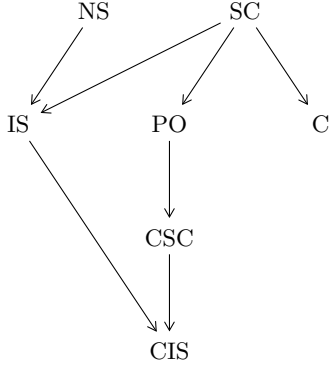


Figure 1: Inclusion relationships between stability concepts. For example, every Nash stable partition is also individually stable.

presented a potential function argument for the existence of a Nash stable partition for ASHG with symmetric preferences. However there are no known polynomial-time algorithms to *compute* such partitions and there is evidence that there may not be any polynomial-time algorithm [14]. In this section, we show that a CIS partition can be computed in polynomial time for ASHG. The algorithm is formally described as Algorithm 1.

THEOREM 1. *A CIS partition can be computed in polynomial time.*

PROOF. Our algorithm to compute a CIS partition can be viewed as successively giving a priority token to players to form the best possible coalition among the remaining players or join the best possible coalition which tolerates the player. The basic idea of the algorithm is described informally as follows. Set variable R to N and consider an arbitrary player $a \in R$. Call a the *leader* of the first coalition S_i with $i = 1$. Move any player j such that $v_a(j) > 0$ from R to S_i . Such players are called the *leader's helpers*. Then keep moving any player from R to S_i which is tolerated by all players in S_i and strictly liked by at least one player in S_i . Call such players *needed players*. Now increment i and take another player a from among the remaining players R and check the maximum utility he can get from among R . If this utility is less than the utility which can be obtained by joining a previously formed coalition in $\{S_1, \dots, S_{i-1}\}$, then send the player to such a coalition where he can get the maximum utility (as long all players in the coalition tolerate the incoming player). Such players are called *latecomers*. Otherwise, form a new coalition S_i around a which is the best possible coalition for player a taking only players from the remaining players R . Repeat the process until all players have been dealt with and $R = \emptyset$. We prove by induction on the number of coalitions formed that no CIS deviation can occur in the resulting partition. The hypothesis is the following:

Consider the k th first formed coalitions S_1, \dots, S_k . Then neither of the following can happen:

1. *There is a CIS deviation by a player from among S_1, \dots, S_k .*
2. *There is a CIS deviation by a player from among $N \setminus \bigcup_{i \in \{1, \dots, k\}} S_i$ to a coalition in $\{S_1, \dots, S_k\}$.*

Input: additively separable hedonic game (N, \mathcal{P}) .

Output: CIS partition.

```

i ← 0
R ← N
while R ≠ ∅ do
  Take any player a ∈ R
  h ← ∑_{b ∈ F(a,R)} v_a(b)
  z ← i + 1
  for k ← 1 to i do
    h' ← ∑_{b ∈ S_k} v_a(b)
    if (h < h') ∧ (∀ b ∈ S_k, v_b(a) = 0) then
      h ← h'
      z ← k
    end if
  end for
  if z ≠ i + 1 then // a is latecomer
    S_z ← {a} ∪ S_z
    R ← R \ {a}
  else // a is leader
    i ← z
    S_i ← {a}
    S_i ← S_i ∪ F(a,R) // add leader's helpers
    R ← R \ S_i
  end if
  while ∃ j ∈ R such that ∀ i ∈ S_z, v_i(j) ≥ 0 and ∃ i ∈ S_z, v_i(j) > 0 do
    R ← R \ {j}
    S_z ← S_z ∪ {j} // add needed players
  end while
end while
return {S_1, ..., S_i}

```

Algorithm 1: CIS partition of an ASHG

Base case.

Consider the coalition S_1 . Then the leader of S_1 has no incentive to leave. The leader's helpers are not allowed to leave because, if they did, the leader's utility would decrease. For each of the needed players, there exists one player in S_1 who does not allow the needed player to leave. Now let us assume a latecomer i arrives in S_1 . This is only possible if the maximum utility that the latecomer can derive from a coalition $C \subseteq (N \setminus S_1)$ is less than $\sum_{j \in S_1} v_i(j)$. Therefore once i joins S_1 , he will only become less happy by leaving S_1 .

Any player $i \in N \setminus S_1$ cannot have a CIS deviation to S_1 . Either i is disliked by at least one player in S_1 or i is disliked by no player in S_1 . In the first case, i cannot deviate to S_1 even he has an incentive to. In the second case, player i has no incentive to move to S_1 because if he had an incentive, he would already have moved to S_1 as a latecomer.

Induction step.

Assume that the hypothesis is true. Then we prove that the same holds for the formed coalitions S_1, \dots, S_k, S_{k+1} . By the hypothesis, we know that players cannot leave coalitions S_1, \dots, S_k . Now consider S_{k+1} . The leader a of S_{k+1} is either not allowed to join one of the coalitions in $\{S_1, \dots, S_k\}$ or if he is, he has no incentive to join it. Player a would already have been member of S_i for some $i \in \{1, \dots, k\}$ if one of the following was true:

- There is some $i \in \{1, \dots, k\}$ such that the leader of S_i

likes a .

- There is some $i \in \{1, \dots, k\}$ such that for all $b \in S_i$, $v_b(a) \geq 0$ and there exists $b \in S_i$ such that $v_b(a) > 0$.
- There is some $i \in \{1, \dots, k\}$, such that for all $b \in S_i$, $v_b(a) = 0$ and $\sum_{b \in S_i} v_a(b) > \sum_{b \in F(i, N \setminus \cup_{i=1}^k S_i)} v_a(b)$ and $\sum_{b \in S_i} v_a(b) \geq \sum_{b \in S_j} v_a(b)$ for all $j \in \{1, \dots, k\}$.

Therefore a has no incentive or is not allowed to move to another S_j for $j \in \{1, \dots, k\}$. Also a will have no incentive to move to any coalition formed after S_1, \dots, S_{k+1} because he can do strictly better in S_{k+1} . Similarly, a 's helpers are not allowed to leave S_{k+1} even if they have an incentive to. Their movement out of S_{k+1} will cause a to become less happy. Also each needed player in S_{k+1} is not allowed to leave because at least one player in S_k likes him. Now consider a latecomer l in S_{k+1} . Latecomer l gets strictly less utility in any coalition $C \subseteq N \setminus \cup_{i=1}^{k+1} S_i$. Therefore l has no incentive to leave S_{k+1} .

Finally, we prove that there exists no player $x \in N \setminus \cup_{j=1}^{k+1} S_j$ such that x has an incentive to and is allowed to join S_i for $i \in \{1, \dots, k+1\}$. By the hypothesis, we already know that x does not have an incentive or is allowed to join a coalition S_i for $i \in \{1, \dots, k\}$. Since x is not a latecomer for S_{k+1} , x either does not have an incentive to join S_{k+1} or is disliked by at least one player in S_{k+1} . \square

Algorithm 1 may also prove useful as a preprocessing or intermediate routine in other algorithms for computing different types of stable partitions of hedonic games.

4. CORE AND STRICT CORE

For ASHG, the problem of testing the core membership of a partition is coNP-complete [20]. This fact does not imply that checking the existence of a core stable partition is NP-hard. Recently, Sung and Dimitrov [22] showed that for ASHG, checking whether a core stable or strict core stable partition exists is NP-hard in the strong sense. Their reduction relied on the asymmetry of the players' preferences. We prove that even with symmetric preferences, checking whether a core stable or a strict core stable partition exists is NP-hard in the strong sense. Symmetry is a natural, but rather strong condition, that can often be exploited algorithmically.

We first present an example of a six-player ASHG with symmetric preferences for which the core (and thereby the strict core) is empty.

EXAMPLE 1. Consider a six player symmetric ASHG adapted from an example by Banerjee et al. [4] where

- $v_1(2) = v_3(4) = v_5(6) = 6$;
- $v_1(6) = v_2(3) = v_4(5) = 5$;
- $v_1(3) = v_3(5) = v_1(5) = 4$;
- $v_1(4) = v_2(5) = v_3(6) = -33$; and
- $v_2(4) = v_2(6) = v_4(6) = -33$

as depicted in Figure 2.

It can be checked that no partition is core stable for the game.

Note that if $v_i(j) = -33$, then i and j cannot be in the same coalition of a core stable partition. Also, players

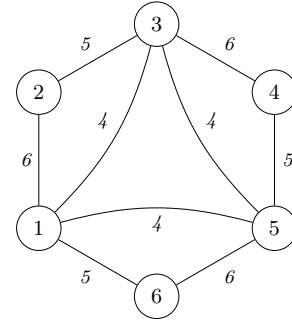


Figure 2: Graphical representation of Example 1. All edges not shown in the figure have weight -33 .

can do better than in a partition of singleton players. Let coalitions which satisfy individual rationality be called feasible coalitions. We note that the following are the feasible coalitions: $\{1, 2\}$, $\{1, 3\}$, $\{1, 5\}$, $\{1, 6\}$, $\{1, 2, 3\}$, $\{1, 3, 5\}$, $\{1, 5, 6\}$, $\{2, 3\}$, $\{3, 4\}$, $\{3, 4, 5\}$, $\{3, 5\}$, $\{4, 5\}$ and $\{5, 6\}$.

Consider partition

$$\pi = \{\{1, 2\}, \{3, 4, 5\}, \{6\}\}.$$

Then,

- $u_\pi(1) = 6$;
- $u_\pi(2) = 6$;
- $u_\pi(3) = 10$;
- $u_\pi(4) = 11$;
- $u_\pi(5) = 9$; and
- $u_\pi(6) = 0$.

Out of the feasible coalitions listed above, the only weakly (and also strongly) blocking coalition is $\{1, 5, 6\}$ in which player 1 gets utility 9, player 5 gets utility 10, and player 6 gets utility 11. We note that the coalition $\{1, 2, 3\}$ is not a weakly or strongly blocking coalition because player 3 gets utility 9 in it. Similarly $\{1, 3, 5\}$ is not a weakly or strongly blocking coalition because both player 3 and player 5 are worse off. One way to prevent the deviation $\{1, 5, 6\}$ is to provide some incentive for player 6 not to deviate with 1 and 5. This idea will be used in the proof of Theorem 2.

We now define a problem that is NP-complete in the strong sense:

Name: EXACTCOVERBY3SETS (E3C):

Instance: A pair (R, S) , where R is a set and S is a collection of subsets of R such that $|R| = 3m$ for some positive integer m and $|s| = 3$ for each $s \in S$.

Question: Is there a sub-collection $S' \subseteq S$ which is a partition of R ?

It is known that E3C remains NP-complete even if each $r \in R$ occurs in at most three members of S [15]. We will use this assumption in the proof of Theorem 2, which will be shown by a reduction from E3C.

THEOREM 2. Checking whether a core stable or a strict core stable partition exists is NP-hard in the strong sense, even when preferences are symmetric.

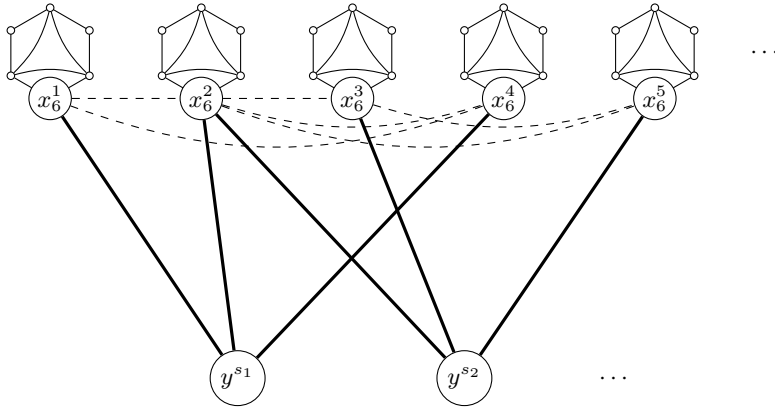


Figure 3: Graphical representation of an ASHG derived from an instance of E3C in the proof of Theorem 2. Symmetric utilities other than -33 are given as edges. Thick edges indicate utility $10\frac{1}{4}$ and dashed edges indicate utility $1/2$. Each hexagon at the top looks like the one in Figure 4.

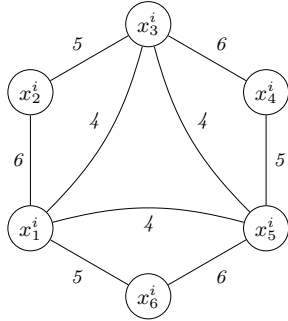


Figure 4: Graphical representation of the ASHG from Example 1 as used in the proof of Theorem 2. All edges not shown in the figure have weight -33 .

PROOF. Let (R, S) be an instance of E3C where $r \in R$ occurs in at most three members of S . We reduce (R, S) to an ASHG with symmetric preferences (N, \mathcal{P}) in which there is a player y^s corresponding to each $s \in S$ and there are six players x_1^r, \dots, x_6^r corresponding to each $r \in R$. These players have preferences over each other in exactly the way players $1, \dots, 6$ have preference over each other as in Example 1.

So, $N = \{x_1^r, \dots, x_6^r \mid r \in R\} \cup \{y^s \mid s \in S\}$. We assume that all preferences are symmetric. The player preferences are as follows:

- For $i \in R$,
 $v_{x_1^i}(x_2^i) = v_{x_3^i}(x_4^i) = v_{x_5^i}(x_6^i) = 6$;
 $v_{x_1^i}(x_6^i) = v_{x_2^i}(x_3^i) = v_{x_4^i}(x_5^i) = 5$; and
 $v_{x_1^i}(x_3^i) = v_{x_3^i}(x_5^i) = v_{x_1^i}(x_5^i) = 4$;
- For any $s = \{k, l, m\} \in S$,
 $v_{x_6^k}(x_6^l) = v_{x_6^l}(x_6^k) = v_{x_6^k}(x_6^m) = v_{x_6^m}(x_6^k) =$
 $v_{x_6^l}(x_6^m) = v_{x_6^m}(x_6^l) = 1/2$; and
 $v_{x_6^k}(y^s) = v_{x_6^l}(y^s) = v_{x_6^m}(y^s) = 10\frac{1}{4}$;
- $v_i(j) = -33$ for any $i, j \in N$ for valuations not defined above.

We prove that (N, \mathcal{P}) has a non-empty strict core (and thereby core) if and only if there exists an $S' \subseteq S$ such that S' is a partition of R .

Assume that there exists an $S' \subseteq S$ such that S' is a partition of R . Then we prove that there exists a strict core stable (and thereby core stable) partition π where π is defined as follows:

$$\begin{aligned} & \{\{x_1^i, x_2^i\}, \{x_3^i, x_4^i, x_5^i\} \mid i \in R\} \cup \{\{y^s\} \mid s \in S \setminus S'\} \\ & \cup \{\{y^s \cup \{x_6^i \mid i \in s\}\} \mid s \in S'\}. \end{aligned}$$

For all $i \in R$,

- $u_\pi(x_1^i) = 6$;
- $u_\pi(x_2^i) = 6$;
- $u_\pi(x_3^i) = 10$;
- $u_\pi(x_4^i) = 11$;
- $u_\pi(x_5^i) = 9$; and
- $u_\pi(x_6^i) = 1/2 + 1/2 + 10\frac{1}{4} = 11\frac{1}{4} > 11$.

Also $u_\pi(y^s) = 3 \times (10\frac{1}{4}) = 30\frac{3}{4}$ for all $s \in S'$ and $u_\pi(y^s) = 0$ for all $s \in S \setminus S'$. We see that for each player, his utility is non-negative. Therefore there is no incentive for any player to deviate and form a singleton coalition. From Example 1 we also know that the only possible strongly blocking (and weakly blocking) coalition is $\{x_1^i, x_5^i, x_6^i\}$ for any $i \in R$. However, x_6^i has no incentive to be part $\{x_1^i, x_5^i, x_6^i\}$ because $u_\pi(x_6^i) = 11$ and $v_{x_6^i}(x_5^i) + v_{x_6^i}(x_1^i) = 6 + 5 = 11$.

Also x_1^i and x_5^i have no incentive to join $\pi(x_6^i)$ because their new utility will become negative because of the presence of the y^s player. Assume for the sake of contradiction that π is not core stable and x_6^i can deviate with a lot of x_6^j s. But, x_6^i can only deviate with a maximum of six other players of type x_6^j because $i \in R$ is present in a maximum of three elements in S . In this case x_6^i gets a maximum utility of only 1. Therefore π is in the strict core (and thereby the core).

We now assume that there exists a partition which is core stable. Then we prove that there exists an $S' \subseteq S$ such that S' is a partition of R . For any $s = \{k, l, m\} \in S$, the new utilities created due to the reduction gadget are only

beneficial to y^s , x_6^k , x_6^l , and x_6^m . We already know that the only way the partition is core stable is if x_6^i can be provided disincentive to deviate with x_5^i and x_1^i . The claim is that each x_6^i needs to be in a coalition with exactly one y^s such that $i \in s \in S$ and exactly two other players x_6^j and x_6^k such that $\{i, j, k\} = s \in S$. We first show that x_6^i needs to be with exactly one y^s such that $i \in s \in S$. Player needs to be with at least one such y^s . If x_6^i is only with other x_6^j s, then we know that x_6^i gets a maximum utility of only $6 \times 1/2 = 3$. Also, player x_6^i cannot be in a coalition with y^s and $y^{s'}$ such that $i \in s$ and $i \in s'$ because both y^s and $y^{s'}$ then get negative utility. Each x_6^i also needs to be with at least 2 other players x_6^j and x_6^k where j and k are also members of s . If x_6^i is with at least three players x_6^j , x_6^k and x_6^l , then there is one element among $a \in \{j, k, l\}$ such that $a \notin s$. Therefore y^s and x_6^a hate each other and the coalition $\{y^s, x_6^i, x_6^j, x_6^k, x_6^l\}$ is not even individually rational. Therefore for the partition to be core stable each x_6^i has to be with exactly one y^s such that $i \in s$ and at least 2 other players x_6^j and x_6^k where j and k are also members of s . This implies that there exists an $S' \subseteq S$ such that S' is a partition of R . \square

5. CONTRACTUAL STRICT CORE AND PARETO OPTIMALITY

In this section, we prove that verifying whether a partition is CSC stable is coNP-complete. Interestingly, coNP-completeness holds even if the partition in question consists of the grand coalition. The proof of Theorem 3 is by a reduction from the following weakly NP-complete problem.

Name: PARTITION

Instance: A set of k positive integer weights $A = \{a_1, \dots, a_k\}$ such that $\sum_{a_i \in A} a_i = W$.

Question: Is it possible to partition A , into two subsets $A_1 \subseteq A$, $A_2 \subseteq A$ so that $A_1 \cap A_2 = \emptyset$ and $A_1 \cup A_2 = A$ and $\sum_{a_i \in A_1} a_i = \sum_{a_i \in A_2} a_i = W/2$?

THEOREM 3. *Verifying whether the partition consisting of the grand coalition is CSC stable is weakly coNP-complete.*

PROOF. The problem is clearly in coNP because a partition π' resulting by a CSC deviation from $\{N\}$ is a succinct certificate that $\{N\}$ is not CSC stable. We prove NP-hardness of deciding whether the grand coalition is *not* CSC stable by a reduction from PARTITION. We can reduce an instance of I of PARTITION to an instance $I' = ((N, \mathcal{P}), \pi)$ where (N, \mathcal{P}) is an ASHG defined in the following way:

- $N = \{x_1, x_2, y_1, y_2, z_1, \dots, z_k\}$,
- $v_{x_1}(y_1) = v_{x_1}(y_2) = v_{x_2}(y_1) = v_{x_2}(y_2) = W/2$,
- $v_{x_1}(z_i) = v_{x_2}(z_i) = a_i$, for all $i \in \{1, \dots, k\}$
- $v_{x_1}(x_2) = v_{x_2}(x_1) = -W$,
- $v_{y_1}(y_2) = v_{y_2}(y_1) = -W$,
- $v_a(b) = 0$ for any $a, b \in N$ for which $v_a(b)$ is not already defined, and
- $\pi = \{N\}$.

We see that $u_\pi(x_1) = u_\pi(x_2) = W$, $u_\pi(y_1) = u_\pi(y_2) = -W$, $u_\pi(z_i) = 0$ for all $i \in \{1, \dots, k\}$. We show that π is not CSC stable if and only if I is a ‘yes’ instance of PARTITION. Assume I is a ‘yes’ instance of PARTITION and there exists an $A_1 \subseteq A$ such that $\sum_{a_i \in A_1} a_i = W/2$. Then, form the partition

$$\pi' = \{\{x_1, y_1\} \cup \{z_i \mid a_i \in A_1\}, \{x_2, y_2\} \cup \{z_i \mid a_i \in N \setminus A_1\}\}.$$

Then,

- $u_{\pi'}(x_1) = u_{\pi'}(x_2) = W$;
- $u_{\pi'}(y_1) = u_{\pi'}(y_2) = 0$; and
- $u_{\pi'}(z_i) = 0$ for all $i \in \{1, \dots, k\}$.

The coalition $C_1 = \{x_1, y_1\} \cup \{z_i \mid a_i \in A_1\}$ can be considered as a coalition which leaves the grand coalition so that all players in N do as well as before and at least one player in C_1 , i.e., y_1 gets strictly more utility. Also, the departure of C_1 does not make any player in $N \setminus C_1$ worse off.

Assume that I is a ‘no’ instance of PARTITION and there exists no $A_1 \subseteq A$ such that $\sum_{a_i \in A_1} a_i = W/2$. We show that no CSC deviation is possible from π . We consider different possibilities for a CSC blocking coalition C :

1. $x_1, x_2, y_1, y_2 \notin C$,
2. $x_1, x_2 \notin C$ and there exists $y \in \{y_1, y_2\}$ such that $y \in C$,
3. $x_1, x_2, y_1, y_2 \in C$,
4. $x_1, x_2 \in C$ and $|C \cap \{y_1, y_2\}| \leq 1$,
5. there exists $x \in \{x_1, x_2\}$ and $y \in \{y_1, y_2\}$ such that $x, y \in C$, $\{x_1, x_2\} \setminus x \notin C$, and $\{y_1, y_2\} \setminus y \notin C$

We show that in each of the cases, C is not a valid CSC blocking coalition.

1. If C is empty, then there exists no CSC blocking coalition. If C is not empty, then x_1 and x_2 gets strictly less utility when a subset of $\{z_1, \dots, z_k\}$ deviates.
2. In this case, both x_1 and x_2 gets strictly less utility when $y \in \{y_1, y_2\}$ leaves N .
3. If $\{z_1, \dots, z_k\} \subset C$, then there is no deviation as $C = N$. If there exists a $z_i \in \{z_1, \dots, z_k\}$ such that $z_i \notin C$, then x_1 and x_2 get strictly less utility than in N .
4. If $|C \cap \{y_1, y_2\}| = 0$, then the utility of no player increases. If $|C \cap \{y_1, y_2\}| = 1$, then the utility of y_1 and y_2 increases but the utility of x_1 and x_2 decreases.
5. Consider $C = \{x, y\} \cup S$ where $S \subseteq \{z_1, \dots, z_k\}$. Without loss of generality, we can assume that $x = x_1$ and $y = y_1$. We know that y_1 and y_2 gets strictly more utility because they are now in different coalitions. Since I is a ‘no’ instance of PARTITION, we know that there exists no S such that $\sum_{a \in S} v_{x_1}(a) = W/2$. If $\sum_{a \in S} v_{x_1}(a) > W/2$, then $u_\pi(x_2) < W$. If $\sum_{a \in S} v_{x_1}(a) < W/2$, then $u_\pi(x_1) < W$.

Thus, if I' is a ‘no’ instance of PARTITION, then there exists no CSC deviation. \square

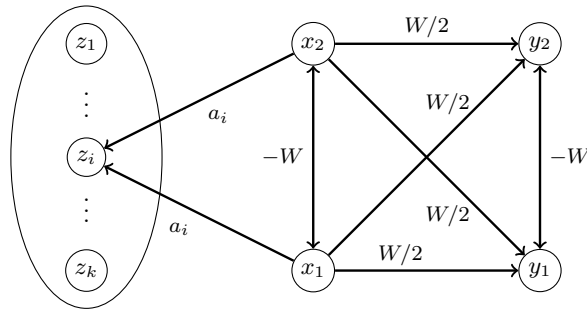


Figure 5: Graphical representation of the ASHG in the proof of Theorem 3. For all $i \in \{1, \dots, k\}$, an edge from x_1 and x_2 to z_i has weight a_i . All other edges not shown in the figure have weight zero.

From the proof of Theorem 3, it can be seen that π is not Pareto optimal if and only if I is a ‘yes’ instance of PARTITION.

THEOREM 4. *Verifying whether the partition consisting of the grand coalition is Pareto optimal is coNP-complete.*

6. CONCLUSION AND DISCUSSION

We presented a number of new computational results concerning stable partitions of ASHG. First, we proposed a polynomial-time algorithm for computing a contractually individually stable (CIS) partition. Secondly, we showed that checking whether the core or strict core exists is NP-hard in the strong sense, even if the preferences of the players are symmetric. Finally, we presented the first complexity result concerning the contractual strict core (CSC), namely that verifying whether a partition is in the CSC is coNP-complete. We saw that considering CSC deviations helps reason about the more complex Pareto optimal improvements. As a result, we established that checking whether the partition consisting of the grand coalition is Pareto optimal is also coNP-complete.

We note that Algorithm 1 may very well return a partition that fails to satisfy individual rationality, i.e., players may get negative utility. It is an open question how to efficiently compute a CIS partition that is guaranteed to satisfy individual rationality. We also note that Theorem 3 may not imply anything about the complexity of computing a CSC partition. Studying the complexity of computing a CSC stable partition is left as future work.

7. ACKNOWLEDGEMENTS

This material is based on work supported by the Deutsche Forschungsgemeinschaft under grants BR-2312/6-1 (within the European Science Foundation’s EUROCORES program LogICCC) and BR 2312/7-1. The authors gratefully acknowledge the support of the TUM’s Faculty Graduate Center CeDoSIA at Technische Universität München.

REFERENCES

- [1] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Princeton University Press, 2009.
- [2] H. Aziz, F. Brandt, and H. G. Seedig. Optimal partitions in additively separable hedonic games. In *Proceedings of the Third International Workshop on Computational Social Choice (COMSOC)*, pages 271–282, 2010.
- [3] C. Ballester. NP-completeness in hedonic games. *Games and Economic Behavior*, 49(1):1–30, 2004.
- [4] S. Banerjee, H. Konishi, and T. Sönmez. Core in a simple coalition formation game. *Social Choice and Welfare*, 18:135–153, 2001.
- [5] S. Barberà, W. Bossert, and P. K. Pattanaik. Ranking sets of objects. In S. Barberà, P. J. Hammond, and C. Seidl, editors, *Handbook of Utility Theory*, volume II, chapter 17, pages 893–977. Kluwer Academic Publishers, 2004.
- [6] A. Bogomolnaia and M. O. Jackson. The stability of hedonic coalition structures. *Games and Economic Behavior*, 38(2):201–230, 2002.
- [7] S. Branzei and K. Larson. Coalitional affinity games and the stability gap. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1319–1320, 2009.
- [8] N. Burani and W. S. Zwicker. Coalition formation games with separable preferences. *Mathematical Social Sciences*, 45(1):27–52, 2003.
- [9] K. Cechlárová. Stable partition problem. In *Encyclopedia of Algorithms*. 2008.
- [10] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 12(2):257–266, 1994.
- [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] J. H. Drèze and J. Greenberg. Hedonic coalitions: Optimality and stability. *Econometrica*, 48(4):987–1003, 1980.
- [13] E. Elkind and M. Wooldridge. Hedonic coalition nets. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 417–424, 2009.
- [14] M. Gairing and R. Savani. Computing stable outcomes in hedonic games. In S. Kontogiannis, E. Koutsoupias, and P. Spirakis, editors, *Proceedings of the 3rd International Symposium on Algorithmic Game Theory (SAGT)*, volume 6386 of *Lecture Notes in Computer Science*, pages 174–185. Springer-Verlag, 2010.
- [15] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

- [16] J. Hajduková. Coalition formation games: A survey. *International Game Theory Review*, 8(4):613–641, 2006.
- [17] M. Olsen. Nash stability in additively separable hedonic games and community structures. *Theory of Computing Systems*, 45(4):917–925, 2009.
- [18] A. Roth and M. A. O. Sotomayor. *Two-Sided Matching: A Study in Game Theoretic Modelling and Analysis*. Cambridge University Press, 1990.
- [19] 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.
- [20] S. C. Sung and D. Dimitrov. On core membership testing for hedonic coalition formation games. *Operations Research Letters*, 35(2):155–158, 2007.
- [21] S. C. Sung and D. Dimitrov. On myopic stability concepts for hedonic games. *Theory and Decision*, 62(1): 31–45, 2007.
- [22] S. C. Sung and D. Dimitrov. Computational complexity in additive hedonic games. *European Journal of Operational Research*, 203(3):635–639, 2010.
- [23] J. von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 2nd edition, 1947.

Complexity of coalition structure generation

Haris Aziz
Department of Informatics
Technische Universität München
85748 Garching bei München, Germany
aziz@in.tum.de

Bart de Keijzer
CWI Amsterdam
1098 XG Amsterdam, The Netherlands
B.de.Keijzer@cwi.nl

ABSTRACT

We revisit the *coalition structure generation problem* in which the goal is to partition the players into exhaustive and disjoint coalitions so as to maximize the social welfare. One of our key results is a general polynomial-time algorithm to solve the problem for all coalitional games provided that player types are known and the number of player types is bounded by a constant. As a corollary, we obtain a polynomial-time algorithm to compute an optimal partition for weighted voting games with a constant number of weight values and for coalitional skill games with a constant number of skills. We also consider well-studied and well-motivated coalitional games defined compactly on combinatorial domains. For these games, we characterize the complexity of computing an optimal coalition structure by presenting polynomial-time algorithms, approximation algorithms, or NP-hardness and inapproximability lower bounds.

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, coordination, and computational complexity

1. INTRODUCTION

Coalition formation is an important issue in multiagent systems with cooperating agents. Coalitional games have been used to model various cooperative settings in operations research, artificial intelligence and multiagent systems (see e.g., [5, 6, 11]). The area of coalitional game theory which studies coalition formation has seen considerable growth over the last few decades. Given a set of agents

Cite as: Complexity of coalition structure generation, Haris Aziz and Bart de Keijzer, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tamer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 191-198. Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

N , a coalitional game is defined by a valuation function $v : N \rightarrow R$ where for $C \subseteq N$, $v(C)$ signifies the value which players in C can generate by cooperating.

In a coalitional game, a partition of the players into exhaustive and disjoint coalitions is called a *coalition structure*. In the *coalition structure generation problem*, the goal is to find a coalition structure π of N that maximizes the social welfare $\sum_{C \in \pi} v(C)$. We will refer to this problem of finding an optimal coalition structure as OPTCS. In this paper, we conduct a detailed investigation of computing optimal coalition structures that give the maximum social welfare. Computing optimal coalition structures is a natural problem in which the aim is to utilize resources in the most efficient manner.

OPTCS has received attention in the artificial intelligence community where the focus has generally been on computing optimal coalition structures for general coalition formation games [16, 20] without any combinatorial structure. Traditionally, the input considered is an oracle called a characteristic function which returns the value for any given coalition (in time polynomial in the number of players). In this setting, it is generally assumed that the value of a coalition does not depend on players who are not in the coalition. Computing optimal coalition structures is a computationally hard task because of the huge number of coalition structures. The total number of coalition structures for a player set of size n is $B_n \sim \Theta(n^n)$ where B_n is the n th Bell number. A number of algorithms have been developed in the last decade which attempt to satisfy many desirable criteria, e.g. outputting an optimal solution or a good approximation, the ability to prune, the anytime property, worst case guarantees, distributed computation etc. [16, 18, 20, 21]. In all of the cases, the algorithms have a worst-case time complexity which is exponential in n . In this paper, we show that the picture is not that bleak if player types are known and the number of player types is bounded by a constant. In fact for such a condition, there is a polynomial-time algorithm for OPTCS for coalitional games. In many multiagent systems, it can be reasonable to assume that the agents can be divided into a bounded number of types according to the player attributes.

We also study the complexity of OPTCS for a number of compact coalitional games. Coalitional games can be represented compactly on combinatorial domains where the valuation function is implicitly defined [9, 10]. Numerous such classes of coalitional games have been the subject of recent research in multiagent systems: weighted voting games [11]; skill games [5]; multiple weighted voting games [4]; network flow games [6]; spanning connectivity games [3]; and match-

ing games [13]. Apart from some exceptions (skill games [7] and marginal contribution nets [17]), most of the algorithmic research for these classes of games has been on computing stability-based solutions. In the paper, we characterize the complexity of OPTCS for many compact games by presenting polynomial-time exact algorithms, approximation algorithms, or NP-hardness and inapproximability lower bounds. Throughout the paper, we assume familiarity with fundamental concepts in computational complexity [1].

Contribution.

In this paper, we undertake a detailed and systematic study of computing optimal coalition structures for many important combinatorial optimization coalitional games.

Our most important result is a general polynomial-time algorithm to compute an optimal coalition structure for any coalitional game when the player types are known and the number of player types is bounded by a fixed constant. As a corollary, we obtain a polynomial-time algorithm to compute an optimal coalition structure for weighted voting games with a constant number of weight values, linear games with a constant number of desirability classes, and all known coalitional skill games with a constant number of skills.

In contrast to our general algorithmic result, we show that finding the player types is intractable in general from a communication and computational complexity point of view.

We present a 2-approximation algorithm for the case of weighted voting games and show that this approximation bound is the best possible. Our approximation and inapproximability results concerning weighted voting games may be of independent interest since they address a problem in the family of knapsack problems [12] which has not been studied before.

We also examine well-known coalitional games based on graphs and characterize the complexity of computing the optimal coalition structures. Interestingly for certain combinatorial optimization games for which the combinatorial optimization problem is NP-hard, the problem of computing an optimal coalition structure is easy.

2. PRELIMINARIES

In this section, we define several important classes of coalitional games and formally define the fundamental computational problem OPTCS.

2.1 Coalitional games

We begin with the formal definition of a *coalitional game*.

DEFINITION 1 (COALITIONAL GAMES). A coalitional game is a pair (N, v) where $N = \{1, \dots, n\}$ is a set of players and $v : 2^N \rightarrow \mathbb{R}$ is a characteristic or valuation function that associates with each coalition $C \subseteq N$ a payoff $v(C)$ where $v(\emptyset) = 0$. A coalitional game (N, v) is monotonic when it satisfies the property that $v(C) \leq v(D)$ if $C \subseteq D$.

Throughout the paper, when we refer to a general coalitional game, we assume such a coalitional game with transferable utility. For the sake of brevity, we will sometimes refer to the game (N, v) as simply v .

DEFINITION 2 (SIMPLE GAME). A simple game is a monotonic coalitional game (N, v) with $v : 2^N \rightarrow \{0, 1\}$ such

that $v(\emptyset) = 0$ and $v(N) = 1$. A coalition $C \subseteq N$ is winning if $v(C) = 1$ and losing if $v(C) = 0$. A minimal winning coalition (MWC) of a simple game v is a winning coalition in which defection of any player makes the coalition losing. A simple game can be represented by (N, W^m) , where W^m is the set of minimal winning coalitions.

For any monotonic coalitional game, one can construct a corresponding threshold game. Threshold versions are common in the multiagent systems literature; see for instance [6, 11].

DEFINITION 3 (THRESHOLD VERSIONS). For each coalitional game (N, v) and each threshold $t \in \mathbb{R}^+$, the corresponding threshold game is defined as the coalitional game (N, v^t) , where

$$v^t(C) = \begin{cases} 1 & \text{if } v(C) \geq t, \\ 0 & \text{otherwise.} \end{cases}$$

It can easily be verified that if a game (N, v) is monotonic, then for any threshold $t \leq v(N)$, the threshold version (N, v^t) is a simple game.

2.2 Coalitional game classes

We now review a number of specific classes of coalitional games. Here we adopt the convention that if CLASS denotes a particular class of games, we have T-CLASS refer to the class of threshold games corresponding to games in CLASS, i.e., for every threshold t , (N, v^t) is in T-CLASS if and only if (N, v) is in CLASS.

Weighted voting games are a widely used class of monotonic games.

DEFINITION 4 (WEIGHTED VOTING GAMES [11]). A weighted voting game (WVG) is a simple game (N, v) for which there is a quota $q \in \mathbb{R}^+$ and a weight $w_i \in \mathbb{R}^+$ for each player i such that

$$v(C) = 1 \text{ if and only if } \sum_{i \in C} w_i \geq q.$$

The WVG with quota q and weights w_1, \dots, w_n for the players is denoted by $[q; w_1, \dots, w_n]$, where we commonly assume $w_i \geq w_{i+1}$ for $1 \leq i < n$.

A multiple weighted voting game (MWVG) is the simple game (N, v) for which there are WVGs $(N, v_1), \dots, (N, v_m)$ such that

$$v(C) = 1 \text{ if and only if } v_k(C) = 1 \text{ for } 1 \leq k \leq m.$$

We denote the MWVG game composed of $(N, v_1), \dots, (N, v_m)$ by $(N, v_1 \wedge \dots \wedge v_m)$.

Other important classes of games are defined on graphs. Among these are *spanning connectivity games*, *independent set games*, *matching games*, *network flow games*, and *graph games*, where either nodes or edges are controlled by players and the value of a coalition of players depends on their ability to connect the graph, enable a bigger flow, or obtain a heavier matching or edge set.

DEFINITION 5 (SPANNING CONNECTIVITY GAME [3]). For each connected undirected graph $G = (V, E)$, we define the spanning connectivity game (SCG) on G as the simple game (N, v) where $N = E$ and for all $C \subseteq E$, $v(C) = 1$ if and only if there exists some $E' \subseteq C$ such that $T = (V, E')$ is a spanning tree.

DEFINITION 6 (INDEPENDENT SET GAME [9]). For each connected undirected graph $G = (V, E)$, we define the independent set game (ISG) on G as the game (N, v) where $N = V$ and for all $C \subseteq V$, $v(C)$ is cardinality of the maximum independent set on the subgraph of G induced on C .

DEFINITION 7 (MATCHING GAME [13]). Let $G = (V, E, w)$ be a weighted graph. The matching game corresponding to G is the coalitional game (N, v) with $N = V$ and for each $C \subseteq N$, the value $v(C)$ equals the weight of the maximum weighted matching of the subgraph induced by C .

Graph games are likewise defined on weighted graphs [10].

DEFINITION 8 (GRAPH GAME [10]). For a weighted graph (V, E, w) , the graph game (GG) is the coalitional game (N, v) where $N = V$ and for $C \subseteq N$, $v(C)$ is the weight of edges in the subgraph induced by C . In this paper, we sometimes assume that the graph corresponding to a graph game has only positive edge weights and denote such graph games by GG^+ . We denote the class of graph games where negative edge weights are allowed by GG . Note that for this latter general class of graph games, we allow the characteristic function v to map to negative reals.

A flow network (V, E, c, s, t) consists of a directed graph (V, E) , with capacity on edges $c : E \rightarrow \mathbb{R}^+$, a source vertex $s \in V$, and a sink vertex $t \in V$. A network flow is a function $f : E \rightarrow \mathbb{R}^+$, which obeys the capacity constraints and the condition that the total flow entering any vertex (other than s and t) equals the total flow leaving the vertex. The value of the flow is the maximum amount flowing out of the source.

DEFINITION 9 (NETWORK FLOW GAME [6]). For a flow network (V, E, c, s, t) , the associated network flow game (NFG) is the coalitional game (N, v) , where $N = E$ and for each $C \subseteq E$ the value $v(C)$ is the value of the maximum flow f with $f(e) = 0$ for all $e \in E \setminus C$.

DEFINITION 10 (PATH COALITIONAL GAMES). For an unweighted directed/undirected graph, $G = (V \cup \{s, t\}, E)$,

- the corresponding Edge Path Coalitional Game (EPCG) is a simple coalitional game (N, v) such that $N = E$ and for any $C \subseteq N$, $v(C) = 1$ if and only if C admits an s - t path.
- the corresponding Vertex Path Coalitional Game (VPCG) is a simple coalitional game (N, v) such that $N = V$ and for any $C \subseteq N$, $v(C) = 1$ if and only if C admits an s - t path.

Finally, we define the class of skill games, which were recently introduced by Bachrach and Rosenschein [5].

DEFINITION 11 (COALITIONAL SKILL GAMES [5]). A coalitional skill domain is composed of players N , a set of tasks $T = \{t_1, \dots, t_m\}$ and a set of skills $S = \{s_1, \dots, s_k\}$. Each player i has a set of skills $S(i) \subseteq S$, and each task t_j requires a set of skills $S(t_j) \subseteq S$. The set of skills a coalition C has is $S(C) = \bigcup_{i \in C} S(i)$. A coalition C can perform task t_j if $S(t_j) \subseteq S(C)$. The set of tasks a coalition C can perform is $T(C) = \{t_j \mid S(t_j) \subseteq S(C)\}$. A task value function is a monotonic function $u : 2^T \rightarrow \mathbb{R}$. A

coalitional skill game (CSG) in a coalitional skill domain is a game (N, v) such that for all $C \subseteq N$, $v(C) = u(T(C))$. A weighted task skill game (WTSG) is a CSG where each task $t_j \in T$ has a weight $w_j \in \mathbb{R}^+$ and the task value function $u(T') = \sum_{j \mid t_j \in T'} w_j$. A threshold version of WTSG can be defined according to Definition 3.

DEFINITION 12 (LINEAR GAMES [23]). On a coalitional game (N, v) , we define the desirability relation \succeq_D as follows: we say that a player $i \in N$ is more desirable than a player $j \in N$ ($i \succeq_D j$) if for all coalitions $C \in N \setminus \{i, j\}$ we have that $v(C \cup \{i\}) \geq v(C \cup \{j\})$. The relations \succ_D (“strictly more desirable”), \sim_D (“equally desirable”), and \preceq_D and \prec_D (“(strictly) less desirable”) are defined in the obvious fashion. Linear games are monotonic simple games with a complete desirability relation, i.e. every pair of players is comparable with respect to \succeq_D . Weighted voting games form a strict subclass of linear games. A linear game on players $N = \{1, \dots, n\}$ is canonical iff $\forall i, j \in N, i < j : i \succeq_D j$. A right-shift of a coalition C is a coalition that can be obtained by a sequence of replacements of players in C by less desirable players. A left-shift of a coalition C is defined analogously. Canonical linear games can be represented by listing their shift-minimal winning coalitions: minimal winning coalitions for which it holds that any right-shift is losing. Similarly they can be represented by listing their shift-maximal losing coalitions, defined as obvious.

2.3 Problem definition

We formally define *coalition structures* and OPTCS.

DEFINITION 13 (OPTIMAL COALITION STRUCTURE). A coalition structure for a game (N, v) is a partition of N . The social welfare attained by a coalition structure π , denoted $v(\pi)$ (we overload notation), is defined as $\sum_{C \in \pi} v(C)$. A coalition structure π is optimal when $v(\pi) \geq v(\pi')$ for every coalition structure π' .

We consider the following standard computational problem in our paper.

DEFINITION 14 (PROBLEM OPTCS). For any class of coalitional games X , and its associated natural representation, the problem $\text{OPTCS}(X)$ is as follows: given a coalitional game $(N, v) \in X$, compute an optimal coalition structure.

3. GAMES WITH FIXED PLAYER TYPES

We study the problem of computing an optimal coalition structure for a coalitional game in the case that the number of *player types* is fixed. Shrot et al. [22] considered player types and showed that some intractable problems become tractable when only dealing with a fixed number of player types. They did not address coalition structure generation in their paper.

DEFINITION 15 (PLAYER TYPE). For a coalitional game (N, v) , we call two players $i, j \in N$ strategically equivalent iff for every coalition $C \in N \setminus \{i, j\}$ it holds that $v(C \cup \{i\}) = v(C \cup \{j\})$. When two players $i, j \in N$ are strategically equivalent, we say that i and j are of the same player type.

DEFINITION 16 (VALID TYPE-PARTITION). A valid type-partition for a game (N, v) is a partition P of N such that for each player set $C \in P$, all players in C are of the same player type.

Let $\text{OPTCS}(k\text{-TYPES})$ be the problem where the goal is to compute an optimal coalition structure for a coalitional game (N, v) , given as input a partition P of N with $|P| \leq k$ and the characteristic function v . Note that if all players are different, then $|P| = n$. In general it is not easy to verify that a given partition for a simple game is a valid type-partition. But under the assumption that we are given a valid type-partition, and v is easy to compute, it turns out that an optimal coalition structure can be computed in polynomial time.

3.1 A general algorithm

Now we will show that there exists a general polynomial-time algorithm to compute an optimal coalition structure for any coalitional game when we are given a valid type-partition with a number of player types bounded by a constant. Our algorithm utilizes dynamic programming to compute an optimal coalition structure provided there are a constant number of player types.

THEOREM 1. *There is a polynomial-time algorithm for $\text{OPTCS}(k\text{-TYPES})$, provided that querying v takes at most polynomial time, and the given input partition is a valid type-partition.*

PROOF. Let $N = \{1, \dots, n\}$ be the player set and $P = \{T_1, \dots, T_k\}$ be the input type-partition. We define *coalition-types* as follows: for non-negative integers t_1, \dots, t_k , the *coalition-type* $T(t_1, \dots, t_k)$ is the set of coalitions $\{C \mid \forall i \in \{1, \dots, k\} : |C \cap T_i| = t_i\}$. In words, coalitions in coalition-type $T(t_1, \dots, t_k)$ have t_i players of type T_i , for $1 \leq i \leq k$. Note that v maps all coalitions of the same coalition-type to the same value.

First our algorithm computes a table V of values for each coalition type. In order to do this we need to query v at most n^k times, since $1 \leq t_i \leq n$ for all i , $1 \leq i \leq k$. Let $\text{time}(v)$ denote the time it takes to query v , then computing V takes $O(n^k \cdot \text{time}(v))$ time.

We proceed with a dynamic programming approach in order to find an optimal coalition structure: Let $f(a_1, \dots, a_k)$ be the optimal social welfare attained by an optimal coalition structure on a game (N', v) with $N' \in \{N' \mid \forall i \in \{1, \dots, k\} : |N' \cap T_i| = a_i\}$. Note that it does not matter which N' we choose from this set: the choice of N' has no effect on the optimal social welfare since all N' are of the same coalition-type. We are interested in computing $f(|T_1|, \dots, |T_k|)$. By $\gamma(G)$, we signify those type-partitions which generate the same total utility as the empty set.

Since $v(\emptyset) = 0$, the following recursive definition of $f(a_1, \dots, a_k)$ follows:

$$f(a_1, \dots, a_k) = \begin{cases} 0 & \text{if } a_i = 0 \text{ for } 1 \leq i \leq k, \\ \max\{f(a_1 - b_1, \dots, a_1 - b_k) + v(b_1, \dots, b_k) \\ \mid \forall i \in \{1, \dots, k\} : b_i \leq a_i\} & \text{otherwise.} \end{cases} \quad (1)$$

The recursive definition of $f(a_1, \dots, a_k)$ directly implies a dynamic programming algorithm. The dynamic programming approach works by filling in a $|T_1| \times \dots \times |T_k|$ table Q , where the value of $f(a_1, \dots, a_k)$ is stored at entry $Q[a_1, \dots, a_k]$. Once the table has been computed,

$f(|T_1|, \dots, |T_k|)$ is returned. The entries of Q are filled in according to (1). In order to utilize (1), “lower” entries are filled in first, i.e. $Q[a_1, \dots, a_k]$ is filled in before $Q[a'_1, \dots, a'_k]$ if $a_i \leq a'_i$ for $1 \leq i \leq k$. Evaluating (1) then takes $O(n^k)$ time (due to the “otherwise”-case of (1), where the maximum of a set of at most n^k elements needs to be computed). There are $O(n^k)$ entries to be computed, so the algorithm runs in $O(n^k \cdot \text{time}(v) + n^{2k})$ time.

It is straightforward to extend this algorithm so that it (instead of outputting only the optimal social welfare) also computes and outputs an actual coalition structure that attains the optimal social welfare. To do so, maintain another table $|T_1| \times \dots \times |T_k|$ table R . At each point in time that some entry of Q is computed, say $Q[a_1, \dots, a_k]$, now we also fill in $R[a_1, \dots, a_k]$. $R[a_1, \dots, a_k]$ contains a description of a set \mathcal{C} of coalitions such that $\sum_{C \in \mathcal{C}} v(C) = f(a_1, \dots, a_k)$ and $\bigcup C \in T(a_1, \dots, a_k)$. It suffices to describe \mathcal{C} by simply listing the type of each $C \in \mathcal{C}$, and it is straightforward to verify that we can set $R(a_1, \dots, a_k)$ to \emptyset if $(a_1, \dots, a_k) \in \gamma(G)$, and otherwise we set $R(a_1, \dots, a_k)$ to $(P(a_1 - b_1, \dots, a_1 - b_k), (b_1, \dots, b_k))$, where (b_1, \dots, b_k) is the argument in the max-expression of (1). \square

3.2 Difficulty of finding types

The polynomial-time algorithm given in Theorem 1 relies on the promise that the type-partition given in the input is valid. A natural question is now whether it is also possible to efficiently compute the type-partition of a game in polynomial time when given only the weaker promise that the number of player types is constant k . We answer this question negatively. For randomized algorithms, we show high communication complexity is necessary, i.e. we show that an exponential amount of information is needed from the characteristic function v when we are given no information on the structure of the characteristic function and we rely only on querying v . In fact, the theorem states that this is the case even when v is simple and $k = 2$. It should be noted that this result also holds for deterministic algorithms, since they are a special case of randomized algorithms. Despite this negative result, we show in Section 3.3 that we can do better for some subclasses of games, when we are provided information on the structure of function v .

THEOREM 2. *Any randomized algorithm that computes a player type-partition when given as input a monotonic simple game (N, v) that has 2 player types, requires at least $\Theta(\frac{2^n}{\sqrt{n}})$ queries to v .*

PROOF. We use Yao’s minimax principle [24], which states that the expected cost of a randomized algorithm on a given problem’s worst-case instances is at least the lowest expected cost among all deterministic algorithms that run on any fixed probability distribution over the problem instances.

Consider the following distribution over the input, where the player set is $N = \{1, \dots, n\}$ and n is even, the number of player types is always $k = 2$, and the given game (N, v) is simple and monotonic. Valuation v is drawn uniformly at random from the set $V = \{v_C \mid C \subset N, |C| = n/2\}$ where in v_C , we call C the *critical coalition*. Function v_C is specified as follows:

- $v_C(D) = 0$ when $|D| < n/2$;
- $v_C(D) = 1$ when $|D| > n/2$;

- $v_C(D) = 1$ when $D = C$, i.e. D is the critical coalition;
- $v_C(D) = 0$ otherwise.

Observe that there are exactly two player types in any instance that has non-zero probability of being drawn under this distribution: when v_C is drawn, the type-partition is $(C, N \setminus C)$. Also observe that for coalitions C of size $\frac{n}{2}$, $v(C) = 1$ with probability $\frac{1}{\binom{n}{n/2}}$, because v is drawn uniformly at random from V .

Now let us consider an arbitrary deterministic algorithm A that computes the type-partition for instances in this input distribution by queries to v . Let C be the critical coalition of $n/2$ players such that $v(C) = 1$. A will have to query $v(C)$ in order to know which characteristic function from V has been drawn, and thus determine the type-partition correctly. Let $Q(v)$ be the sequence of queries to v that A generates. Let $Q'(v)$ be the subsequence obtained by removing from $Q(v)$ all queries $v(D)$ such that $|D| \neq n/2$ and all queries that occur after $v(C)$. Because A is deterministic, the query sequence of A is the same among all instances up to querying the critical coalition, since the critical coalitions are the only points in which the characteristic functions of V differ from each other. Therefore the expected length of $Q'(v)$ is $\binom{n}{n/2}/2$. Because A was chosen arbitrarily, we conclude that also the most efficient deterministic algorithm is expected to make at least $\binom{n}{n/2}/2 = \Theta(\frac{2^n}{\sqrt{n}})$ queries to v , and the theorem now follows from Yao's principle. \square

Shrot et al. [22] showed that checking whether two players are of the same type is NP-hard for coalitional games defined by Conitzer and Sandholm [8]. But the games are such that even computing the value of a coalition is NP-hard. One can say something stronger.

PROPOSITION 1. *There exists a representation of coalitional games for which checking whether two players are of the same type is coNP-complete even if the value of each coalition can be computed in polynomial time.*

PROOF. A coalition $C \subseteq N \setminus \{i, j\}$ such that $v(C \cup \{i\}) \neq v(D \cup \{j\})$ is a polynomial-time certificate for membership in coNP. Also, it is well known that checking whether two players in a WVG have the same Banzhaf index is coNP-complete [15]. Since two players in a WVG are of the same type if and only if they have same the Banzhaf index, we are done. \square

3.3 Applications of Theorem 1

Theorem 2 and Proposition 1 indicate that finding player types is in general a difficult task. Despite these negative results, Theorem 1 still applies to all classes of coalitional games and many natural settings where the type-partition is implicitly or explicitly evident:

COROLLARY 1. *There exists a polynomial-time algorithm that solves OPTCS(WVG) in the following cases: 1.) in the input game (given in weighted form), the number of distinct weights is constant; 2.) in the input game (given in weighted form) the number of distinct weight vectors for the players is constant.*

PROOF. When two players have the same weight (in the case of WVGs) or weight vectors (in the case of MWVGs), they are strategically equivalent. Therefore we can type-partition the players according to their weights and apply Theorem 1. \square

There exists a polynomial-time algorithm for computing the desirability classes, when given the list of shift-minimal winning coalitions of a linear game [2]. This immediately yields the following corollary:

COROLLARY 2. *In the following cases, there exists a polynomial-time algorithm that computes an optimal coalition structure for linear games with a constant number of desirability classes: 1.) the input game is represented as a list of (shift-)minimal winning coalitions; 2.) the input game is represented as a list of (shift-)maximal losing coalitions;*

Bachrach et al. [7] proved that OPTCS(CSG) is polynomial-time solvable if the number of tasks is constant and the 'skill graph' has bounded tree-width. As a corollary of Theorem 1, we obtain a complementing positive result which applies to all of the coalitional skill games defined in [5].

COROLLARY 3. *There exists a polynomial-time algorithm that computes an optimal coalition structure for WTSGs and T-WTSGs with at most a fixed number of player types or a fixed number of skills.*

PROOF. Assume that there the number of skills is a constant k' . Then there is a maximum of $2^{k'}$ player types. A polynomial-time algorithm that computes an optimal coalition structure now follows from Theorem 1. \square

4. WEIGHTED VOTING GAMES AND SIMPLE GAMES

In this section, we examine weighted voting games (WVGs) and, more generally, simple games. Weighted voting games are coalitional games widely used in multiagent systems and AI. We have already seen that there exists a polynomial-time algorithm to compute an optimal coalition structure for WVGs with a constant number of weight values. We show that if the number of weight values is not a constant, then the problem becomes strongly NP-hard.

PROPOSITION 2. *For a WVG, checking whether there is a coalition structure that attains social welfare k or more is NP-complete.*

PROOF. We prove this by a reduction from an instance of the classical NP-hard PARTITION problem to checking whether a coalition structure in a WVG gets social welfare at least 2. An instance of the problem k -PARTITION is a set of n integer weights $A = \{a_1, \dots, a_n\}$ and the question is whether it is possible to partition A , into k subsets $P_1 \subseteq A, \dots, P_k \subseteq A$ such that $P_i \cap P_j = \emptyset$ and $\bigcup_{1 \leq i \leq k} P_i = A$ and for all $i \in \{1, \dots, k\}$, $\sum_{a_j \in A_i} a_j = \sum_{1 \leq j \leq n} a_j / k$.

Without loss of generality, assume that $W = \sum_{a_i \in A} a_i$ is a multiple of k . Given an instance of k -PARTITION $I = \{a_1, \dots, a_k\}$, we can transform it to a WVG $v = [q; w_1, \dots, w_k]$ where $w_i = a_i$ for all $i \in \{1, \dots, k\}$ and $q = W/k$. Then the answer to I is yes if and only if there exists a coalition structure π for v such that $v(\pi) = k$. \square

Since 3-PARTITION is strongly NP-complete, it follows that OPTCS(WVG) is strongly NP-hard. This is contrary to the other results concerning WVGs where computation becomes easy when the weights are encoded in unary [15]. Note that any strongly NP-hard optimization problem with a polynomially bounded objective function cannot have an FPTAS

unless $P = NP$. Proposition 2 does not discourage us from seeking an approximation algorithm for WVGs. We show that there exists a 2-optimal polynomial-time approximation algorithm:

PROPOSITION 3. *There exists a 2-optimal polynomial-time approximation algorithm for OPTCS(WVG).*

PROOF. Consider the following algorithm: Let $[q; w_1, \dots, w_n]$ be the input (so $N = \{1, \dots, n\}$). We assume without loss of generality that $w_i \leq q$ for all i . The algorithm first sets $p[0] := 0$, and then computes for some number c the values $p[1], \dots, p[c]$ using the rule

$$p[i] := \begin{cases} n & \text{if } \sum_{k=p[i-1]+1}^n w_k < q, \\ \min\{j \mid \sum_{k=p[i-1]+1}^j w_k \geq q, \\ (p[i-1] + 1) \leq j \leq n\} & \text{otherwise,} \end{cases} \quad (2)$$

where c is taken such that $p[c] = n$. The algorithm outputs the coalition structure $\{C_1, \dots, C_c\}$, where for $1 \leq i \leq c$, $C_i = \{p[i-1] + 1, \dots, p[i]\}$.

Observe that the coalitions C_1 to C_{c-1} are all winning and C_c is not necessarily winning, so the value of the computed coalition structure is at least $c-1$. By our assumption, the total weight of any of the coalitions C_1, \dots, C_{c-1} is less than $2q$, and the total weight of C_c is less than q . Therefore, the total weight of N is strictly less than $q(2c-1)$, so the optimal social welfare is at most $2c-2 = 2(c-1)$. This is two times the social welfare of the coalition structure computed by the algorithm. \square

A tight example for the algorithm described in the proof of Theorem 3 would be $[q; q-\epsilon, q-\epsilon, \epsilon, \epsilon]$, where q is a fixed constant and ϵ is any positive real number strictly less than $q/2$. On this input, the algorithm outputs a coalition structure that attains a social welfare of 1, while the optimal social welfare is clearly 2. The following proposition shows that there does not exist a better polynomial-time approximation algorithm under the assumption that $P \neq NP$.

PROPOSITION 4. *Unless $P = NP$, there exists no polynomial-time algorithm which computes an α -optimal coalition structure for a WVG where $\alpha < 2$.*

PROOF. We would be able to solve the NP-complete problem PARTITION in polynomial time if there existed a (< 2)-optimal polynomial-time approximation algorithm for OPTCS(WVG). We could reduce a partition instance (w_1, \dots, w_n) to a weighted voting game $[q; w_1, \dots, w_n]$ where $q = \frac{\sum_{i=1}^n w_i}{2}$. Because the sum of all weights of the players is $2q$, a (< 2)-optimal approximation algorithm would output an optimal coalition structure when provided with this instance. The output coalition structure directly corresponds to a solution of the original PARTITION instance, in case it exists. Otherwise, the social welfare attained by the output coalition structure is 1. \square

Simple games that are not necessarily weighted, and are represented by the list of minimal winning coalitions, are even harder to approximate.

PROPOSITION 5. OPTCS(MWC), *i.e.* OPTCS for simple games represented as a list of minimal winning coalitions, cannot be approximated within any constant factor unless $P = NP$.

PROOF. This can be proved by a reduction from an instance of the classical NP-hard maximum clique (MAX-CLIQUE) problem. It is known that MAXCLIQUE cannot be approximated within any constant factor [14].

Consider the instance I of MAXCLIQUE represented by an undirected graph $G_I = (V, E)$. Transform I into instance $I' = (N, W^m)$ of OPTCS(MWC) in the following way. Define $N = \{\{v, v'\} : v \in V, v' \in V\}$ to be all subsets of V of cardinality 2. Next, set $W^m = \{C_i : i \in V\}$, and for all $i \in V$ define $C_i = \{\{i, j\} \mid \{i, j\} \notin E\}$. Now two coalitions C_i and C_j are disjoint if and only if $\{i, j\} \in E$. Then the maximum clique size is greater than or equal to k if and only if there is a coalition structure for (N, W^m) that attains social welfare k . Now assume that there exists a polynomial-time algorithm which computes a coalition structure π which gets social welfare within a constant factor α of the maximum possible social welfare k . Then we can use π to get a constant-factor approximation solution to instance I in polynomial time in the following way. Consider the set of vertices $\{i : C_i \in \pi\}$. Since for $C_i, C_j \in \pi$, C_i and C_j are disjoint, then we know that $(i, j) \in E$. Therefore the vertices $\{i : C_i \in \pi\}$ form a clique of size k/α . \square

5. GAMES ON GRAPHS

Numerous classes of coalitional games are based on graphs. We characterize the complexity of OPTCS for many of these classes in the section. We first turn our attention to one such class for which the computation of cooperative game solutions is well studied [10]. We see that that OPTCS is computationally hard in general for graph games:

PROPOSITION 6. *For the general class of graph games GG , the problem OPTCS is strongly NP-hard.*

PROOF. We prove by presenting a reduction from the strongly NP-hard problem MAXCUT. Consider an instance I of MAXCUT with a connected undirected graph $G = (V, E, w)$ and non-negative weights $w(i, j)$ for each edge (i, j) . Let $W = \sum_{(i, j) \in E} w(i, j)$ and define $P(i)$ as the vertices on the same side as vertex i . We show that if there is a polynomial-time algorithm which computes an optimal coalition structure, then we have a polynomial-time algorithm for MAXCUT. There exists a polynomial-time reduction that reduces I to an instance $I' = (V', E', w')$ of OPTCS for graph games where $V' = V \cup \{x_1, x_2\}$ and $E' = E \cup \{\{x_1, i\} : i \in N\} \cup \{\{x_2, i\} : i \in N\} \cup \{\{x_1, x_2\}\}$. The weight function w' is defined as follows: $w'(a, b) = -w(a, b)$ if $a, b \in V$, $w'(a, b) = W + 1$ if $a \in \{x_1, x_2\}$ and $b \in V$, $w'(a, b) = -(|V| + 1)W$ if $a = x_1$ and $b = x_2$.

We now show that a solution to instance I' of OPTCS(GG) can be used to solve instance I of MAXCUT. Assume that π' is an optimal coalition structure for I' . Then we know that π' is of the form $\{\{x_1, A'\}, \{x_2, B'\}\}$ where (A', B') is a partition of V . We also know that $\sum_{a \notin \pi'(b)} w'(a, b)$ is minimized in π' . Therefore, we have a corresponding partition π of V such that $\sum_{a \notin \pi(b)} w(a, b)$ is maximized. \square

OBSERVATION 1. *It is clear that for GG^+ , the coalition structure containing only the grand coalition is the optimal coalition structure.*

We now present some positive results concerning OPTCS for other games on graphs:

PROPOSITION 7. $\text{OPTCS}(\text{SCG})$ can be solved in polynomial time.

PROOF. For a SCG, OPTCS is equivalent to computing the maximum number of edge disjoint spanning subgraphs. Clearly, the maximum number of edge disjoint spanning trees is greater than or equal to the maximum number of spanning subgraphs. Since the spanning trees are also spanning subgraphs, the problem reduces to computing the maximum number of disjoint spanning trees. The problem is solvable in $O(m^2)$ [19]. \square

PROPOSITION 8. For EPCGs and VPCGs , OPTCS can be solved in polynomial time.

PROOF. The problems are equivalent to computing the maximum number of edge disjoint and vertex disjoint s - t paths respectively. There are well-known algorithms to compute them. For example, the maximum number of edge-disjoint s - t paths is equal to the max flow value of the graph in which each edge has unit capacity. The problem of maximizing the number of vertex disjoint paths can be reduced to maximizing the number of vertex disjoint paths in the following way: duplicate each vertex (apart from s and t) with one getting all ingoing edges, and the other getting all the outgoing edges, and an internal edge between them with the node weight as the edge weight. \square

PROPOSITION 9. The coalition structure containing only the grand coalition is an optimal coalition structure for: 1.) NFGs and 2.) Matching games.

PROOF. 1.) Assume there is a coalition structure π of the edges which achieves the total social welfare of s . This means that the sum of the net flow for each $E' \in \pi$ totals s . Since each member of π is mutually exclusive, for any $A, B \in \pi$, the flows in A and B do not interact with each other. Now, consider the coalition structure $\pi' = \{E\}$ which consists of the grand coalition. Then E can achieve a network flow of at least s by having exactly the same flows as that of π , we know that $v(\pi') \geq s$. Therefore, the coalition structure consisting of only the grand coalition attains a social welfare that is at least the social welfare attained by any other coalition structure.

2.) Assume there is a coalition structure $\pi = \{V_1, \dots, V_k\}$ of the vertices that attains a social welfare of s . Let the maximum weighted matching of the graph $G[V_i]$ restricted to vertices V_i be m_i . Then we know that $\sum_{1 \leq i \leq k} m_i = s$. Since each member of π is mutually exclusive, for any $V_i, V_j \in \pi$, the matchings in $G(V_i)$ and $G(V_j)$ have no intersection with each other. Now, consider the coalition structure $\pi' = \{E\}$ which consists of the grand coalition. Then V can achieve a maximum matching of at least s by having exactly the same matchings as that of vertex sets in π . This implies that $v(\pi') \geq s$. Therefore, the coalition structure consisting of only the grand coalition attains a social welfare that is at least the social welfare attained by any other coalition structure. \square

On the other hand, the threshold versions of certain games are computationally harder to solve because of their similarity to WVGs [4]. As a corollary of Prop. 4, we obtain the following:

COROLLARY 4. Unless $\text{P} = \text{NP}$, there exists no polynomial-time algorithm which computes an α -optimal

coalition structure for $\alpha < 2$ and for the following classes of games: 1. T-NFG . 2. T-Matching game and 3. T-GG^+ .

In some cases, OPTCS may be expected to be intractable because the coalitional game is defined on a combinatorial optimization domain which itself is intractable. We observe that even if computing the value of coalitions is intractable, solving OPTCS may be easy:

OBSERVATION 2. Given an instance of maximum independent set, graph $G = (V, E)$, finding the value of the coalition $v(N)$ is NP-hard , but the optimal coalition structure is all singletons.

6. CONCLUSIONS

Coalition structure generation is an active area of research in multiagent systems. We presented a general positive algorithmic result for coalition structure generation, namely that an optimal coalition structure can be computed in polynomial time if the player types are known and the number of player types is bounded by a constant. In many large multiagent systems, it is a valid assumption that there are a lot of agents but the agents can be divided into a bounded number of strategic classes. For example, skill games are well motivated for coordinated rescue operation settings [5, 7]. In these settings, there may be a large number of rescuers but they can be divided into a constant number of types such as firemen, policemen and medics. We have also undertaken a detailed study of the complexity of computing an optimal coalition structure for a number of well-studied games and well-motivated games in AI, multiagent systems and operations research. The results are summarized in Table 1.

7. ACKNOWLEDGEMENTS

This material is based on work supported by the Deutsche Forschungsgemeinschaft under grants BR-2312/6-1 (within the European Science Foundation's EUROCORES program LogICCC) and BR 2312/7-1. We also thank Hans Georg Seedig and the anonymous referees for helpful feedback.

REFERENCES

- [1] S. Arora and B. Barak. *Computational Complexity*. Cambridge University Press, 2009.
- [2] H. Aziz. Complexity of comparison of influence of players in simple games. In *Proceedings of the Second International Workshop on Computational Social Choice (COMSOC 2008)*, pages 61–72, 2008.
- [3] H. Aziz, O. Lachish, M. Paterson, and R. Savani. Wiretapping a hidden network. In *Proceedings of the 5th International Workshop on Internet and Network Economics (WINE)*, pages 438–446, 2009.
- [4] H. Aziz, F. Brandt, and P. Harrenstein. Monotone cooperative games and their threshold versions. In *Proceedings of the 9th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1017–1024, 2010.
- [5] Y. Bachrach and J. S. Rosenschein. Coalitional skill games. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1023–1030, 2008.

Game class	Complexity of OPTCS
Coalition value oracle (valid type-partition with a const. no. of player types)	P (Th. 1)
WVG (const no. weight values)	P (Cor. 1)
(T-)WTCSGs (const. no. of skills or const. no. of player types)	P (Cor. 3)
WCSG (const. tasks, bounded tree-width skill graph)	P [7]
SCG	P (Prop. 7)
EPCG and VPCG	P (Prop. 8)
NFG and Matching Game	P (Prop. 9)
Marginal Contribution Nets	NP-hard [17]
GG ⁺	P (Obs. 1)
Independent Set Game	P (Obs. 2)
GG	Strongly NP-hard (Prop. 6)
(N, W^m)	NP-hard to approx. within const. factor (Prop. 5)
WVG	Strongly NP-hard (Prop. 2);
	NP-hard to approx. within factor < 2 (Prop. 4)
T-Matching; T-NFG; T-GG	NP-hard to approx. within factor < 2 (Cor. 4)
CSG	NP-hard even for SCSGs [7]

Table 1: Summary of complexity results for OPTCS

- [6] Y. Bachrach and J. S. Rosenschein. Power in threshold network flow games. *Journal of Autonomous Agents and Multi-Agent Systems*, 18(1):106–132, 2009.
- [7] Y. Bachrach, R. Meir, K. Jung, and P. Kohli. Coalitional structure generation in skill games. In M. Fox and D. Poole, editors, *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*. AAAI Press, 2010.
- [8] V. Conitzer and T. Sandholm. Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence*, 170:607–619, 2006.
- [9] X. Deng and Q. Fang. Algorithmic cooperative game theory. In A. Chinchuluun, P. M. Pardalos, A. Migdalas, and L. Pitsoulis, editors, *Pareto Optimality, Game Theory And Equilibria*, volume 17 of *Springer Optimization and Its Applications*, pages 159–185. Springer, 2008.
- [10] X. Deng and C. H. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 12(2):257–266, 1994.
- [11] E. Elkind, L. A. Goldberg, P. W. Goldberg, and M. J. Wooldridge. Computational complexity of weighted threshold games. In *Proceedings of the 22nd AAAI Conference on Artificial Intelligence (AAAI)*, pages 718–723. AAAI Press, 2007.
- [12] H. Kellerer, U. Pferschy, and D. Pisinger. *Knapsack Problems*. Springer, Berlin, Germany, 2004.
- [13] W. Kern and D. Paulusma. Matching games: the least core and the nucleolus. *Mathematics of Operations Research*, 28(2):294–308, 2003.
- [14] F. Maffioli and G. Galbiati. Approximability of hard combinatorial optimization problems: an introduction. *Annals of Operations Research*, 96(1):221–236, 11 2000.
- [15] T. Matsui and Y. Matsui. A survey of algorithms for calculating power indices of weighted majority games. *Journal of the Operations Research Society of Japan*, 43(1):71–86, 2000.
- [16] T. Michalak, J. Sroka, T. Rahwan, M. Wooldridge, P. Mcburney, and N. Jennings. A distributed algorithm for anytime coalition structure generation. In *Proceedings of the 9th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2010.
- [17] N. Ohta, V. Conitzer, R. Ichimura, Y. Sakurai, A. Iwasaki, and M. Yokoo. Coalition structure generation utilizing compact characteristic function representations. In *15th International Conference on the Principles and Practice of Constraint Programming (CP)*, pages 623–638, 2009.
- [18] T. Rahwan, S. Ramchurn, N. Jennings, and A. Giovannucci. An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research (JAIR)*, 34:521–567, 2009.
- [19] J. Roskind and R. E. Tarjan. A Note on Finding Minimum-Cost Edge-Disjoint Spanning Trees. *Mathematics of Operations Research*, 10(4):701–708, 1985.
- [20] 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.
- [21] T. Service and J. Adams. Approximate Coalition Structure Generation. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*, 2010.
- [22] T. Shrot, Y. Aumann, and S. Kraus. On agent types in coalition formation problems. In *Proceedings of the 9th International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2010.
- [23] A. D. Taylor and W. S. Zwicker. *Simple Games*. Princeton University Press, 1999.
- [24] A. C.-C. Yao. Probabilistic computations: Toward a unified measure of complexity. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 222–227, Washington, DC, USA, 1977. IEEE Computer Society.

Equilibrium Approximation in Simulation–Based Extensive–Form Games

Nicola Gatti
Politecnico di Milano
Piazza Leonardo da Vinci 32
Milano, Italy
ngatti@elet.polimi.it

Marcello Restelli
Politecnico di Milano
Piazza Leonardo da Vinci 32
Milano, Italy
restelli@elet.polimi.it

ABSTRACT

The class of *simulation-based games*, in which the payoffs are generated as an output of a simulation process, recently received a lot of attention in literature. In this paper, we extend such class to games in extensive form with continuous actions and perfect information. We design two convergent algorithms to find an approximate subgame perfect equilibrium (SPE) and an approximate Nash equilibrium (NE) respectively. Our algorithms can exploit different optimization techniques. In particular, we use: *simulated annealing*, *cross entropy* method, and *Lipschitz optimization*. We produce an extensive experimental evaluation of the performance of our algorithms in terms of approximation degree of the optimal solution and number of evaluated samples. Finding approximate NE and SPE requires exponential time in the game tree depth: an SPE can be computed in game trees with a small depth, while the computation of an NE is easier.

Categories and Subject Descriptors

I.2.11 [Computing Methodologies]: Distributed Artificial Intelligence.

General Terms

Algorithms, Economics.

Keywords

Game Theory (cooperative and non-cooperative).

1. INTRODUCTION

Non-cooperative game theory provides formal tools to model situations wherein rational agents interact and describes the pertinent solution concepts [5]. The central solution concepts are the Nash equilibrium for games in which agents play simultaneously (said in *strategic form*) and the subgame perfect equilibrium when agents play sequentially (said in *extensive form*). Game theory proves that any finite game admits at least an equilibrium (Nash and subgame perfect), however it leaves open the problem to compute it. Equilibrium computation is currently one of the most challenging problem in computer science [17].

Cite as: Equilibrium Approximation in Simulation–Based Extensive–Form Games, Gatti, Restelli, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 199–206.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

The formal model of a game is based on the concept of *mechanism*. It defines the rules of the game, specifying the number of roles of the agents, the actions available to the agents, the sequential structure of the game, and the preferences of the agents over the outcomes (usually expressed as utility functions). Almost the entire game theory deals with games where the agents' utility functions are known in analytical form. Recently, the class of *simulation-based games* have been proposed, in which the agents' utility are not analytical known, but they are the result of a (usually continuous) simulation process. The main interest in studying these games lays in developing algorithms able to find agents' (approximate) equilibrium strategies without having any information about the simulation process.

The main work on equilibrium computation for simulation-based games is described in [21]. The authors provide algorithms based on best response iteration to find an approximate Nash equilibrium, where the best responses are approximated by using stochastic optimization techniques (precisely, simulated annealing). The authors discuss also the conditions that assure their algorithms to converge (in probability) to an equilibrium. However, this work is applicable only to games in strategic form. The study of simulation-based games in extensive form has not received enough attention. To the best of our knowledge, the unique pertinent result is provided in [20], where the authors employ the simulation-based framework for mechanism design.

In this paper, we provide the first study of simulation-based (continuous) extensive-form games. After having defined the class of simulation-based extensive-form games (Section 2), we provide two algorithms to compute an approximate Nash equilibrium (NE) and an approximate subgame perfect (SPE) respectively, that work directly on the game tree (Section 3). These algorithms exploit black-box (stochastic) optimization techniques. We develop our algorithms with three different optimization techniques: simulated annealing (as in [21]), cross entropy method, and Lipschitz optimization. Then, we experimentally evaluate the performance of our algorithms (and optimization techniques) in terms of: ϵ value of the found ϵ -approximate equilibrium, number of evaluated samples, and time (Section 5). We experimentally show that an NE can be found in game trees deeper than in the SPE case.

2. SIMULATION-BASED AND EXTENSIVE-FORM GAMES

The class of simulation-based games was introduced in [21] and captures situations where the agents' utility functions

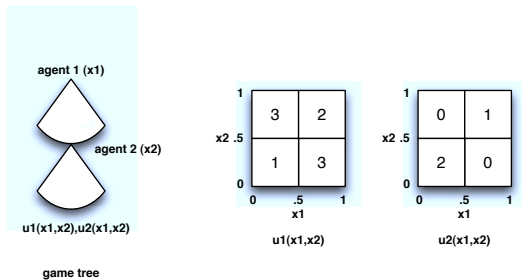


Figure 1: A two-level continuous game with two agents: game tree (left), and agents’ utility functions (middle and right).

are not analytically known, but they are given as the result of a simulation process. Formally, there is an oracle \mathcal{O} that, given an outcome of the game, produces a (possibly noisy) sample from the agents’ joint utility functions. Since most of simulation processes are based on real-valued variables, a very interesting class of simulation-based games is that of games where the agents’ actions are continuous. In these games, the actions available to the agents are the assignment of values to one or more real-valued variables. In what follows, we define the concept of simulation-based extensive-form games and we review the appropriate solution concepts. As customary in game theory, we distinguish the *mechanism*, that specifies the game rules, from the *strategies*, that specify the agents’ behavior during the game.

2.1 Mechanism

A finite perfect-information extensive-form game is a tuple $(N, A, V, T, \iota, \rho, \chi, u)$, where: N is the set of n agents, A is a set of actions, V is the set of decision nodes of the game tree, T is the set of terminal nodes of the game tree, $\iota : V \rightarrow N$ is the agent function that specifies the agent that acts at a given decision node, $\rho : V \rightarrow \wp(A)$ returns the actions available to agent $\iota(v)$ at decision node v , $\chi : V \times A \rightarrow V \cup T$ assigns the next (decision or terminal) node to each pair composed of a decision node v and an action a available at v , and $u = (u_1, \dots, u_n)$ is the set of agents’ utility functions where $u_i : T \rightarrow \mathbb{R}$. An extensive-form game is with *imperfect-information* when some action of some agent is not perfectly observable by the agent’s opponents. In this paper, we limit our study to perfect-information games. Furthermore, we focus on games with *perfect recall*, where every agent recalls all the previously undertaken actions.

In our work, we consider *continuous* perfect-information extensive-form games. In these games, sets A , V , and T are compact. Given a decision node v , agent $\iota(v)$ has a continuous set of actions $\subseteq A$. Each action can lead to a different decision or terminal node. (The model can be easily extended to the case in which A , V , and T are mixed sets composed of continuous and discrete elements.)

EXAMPLE 2.1. Consider Figure 1, there are two agents (i.e., agent 1 and agent 2) that play according to a two-level game tree where the first agent to play is agent 1. Agent 1 can assign a real value to x_1 from the range $[0, 1]$. After the assignment by agent 1, agent 2 can assign a real value to x_2 from the range $[0, 1]$. Agents’ utility functions are defined on x_1 and x_2 as shown in figure.

In a simulation-based extensive-form game, players’ utility functions u are not known. Formally, the component u in the tuple $(N, A, V, T, \iota, \rho, \chi, u)$ is substituted by oracle \mathcal{O} whose argument is $t \in T$. We say that $(N, A, V, T, \iota, \rho, \chi, \bar{u})$ where

$\bar{u} = E[\mathcal{O}]$ is the *underlying game* of the simulation-based game where the oracle is \mathcal{O} . Given $u(t)$, we denote by $u_i(t)$ the component of u reporting the utility of agent i .

2.2 Strategies

In an extensive-form game, a *pure strategy* σ_i is a plan of actions specifying one action for each decision node of agent i . A mixed strategy σ_i is a randomization over pure strategies (plans). When the game is continuous, the definition of a single plan is extremely complex and cannot be conveniently used. An alternative and more compact representation is given by *behavioral strategies*. These define the behavior of an agent at each node independently of the choices taken at other nodes. Essentially, a behavioral strategy σ_i assigns each decision node $v \in V$ a probability distribution over the actions available at v . With perfect recall, the two representations (plans and behavioral) are equivalent. A *strategy profile* collects the strategies of all the agents and is defined as $\sigma = (\sigma_1, \dots, \sigma_n)$.

With continuous games, behavioral strategies are usually expressed as functions that map actions played at previous nodes to an action (or a probability distribution) available at the current node.

EXAMPLE 2.2. Consider the game in Fig. 1, possible strategies for agent 1 and agent 2 are:

$$\sigma_1 = \{x_1 = .3\}$$

$$\sigma_2 = \begin{cases} x_2 = .6 & \text{if } x_1 \leq .2 \\ x_2 = .9 & \text{if } x_1 > .2 \end{cases}$$

Solving a game means to find a strategy profile in which agents’ strategies are somehow in equilibrium. Under the assumption that information is complete and common we can define the concept of *Nash equilibrium* (NE) as a strategy profile σ such that for all $i \in N$: σ_i is a best response to σ_{-i} where σ_{-i} is given by σ once σ_i has been removed (easily, a strategy σ_i is a best response when no other strategy provides a larger utility). It is well known that in extensive-form games some Nash equilibria may be not reasonable with respect to the sequential structure of the game [5]. When information is perfect, the appropriate refinement of Nash for extensive-form games is the *subgame perfect equilibrium* (SPE) [5]. With perfect information, a *subgame* is a subtree of the game tree. Obviously, in continuous games, there are infinite subgames. A subgame perfect equilibrium is a strategy profile that is a Nash equilibrium in every subgame. Every finite extensive-form with perfect information has at least one subgame perfect equilibrium in pure strategies [5]. The same result holds when the game is continuous with bounded utility functions [8]. Since the subgame perfect equilibrium is a refinement of the Nash equilibrium, every continuous perfect-information extensive-form game always admits at least one Nash equilibrium (instead, continuous one-shot games may not admit any Nash equilibrium).

EXAMPLE 2.3. Consider the game depicted in Fig. 1. There is ‘essentially’ a unique SPE (rigorously speaking, there are infinite SPEs, but they are all equivalent in terms of utilities), represented in Fig. 2 where:

$$\sigma_1 = \{x_1 > .5\}$$

$$\sigma_2 = \begin{cases} x_2 \leq .5 & \text{if } x_1 \leq .5 \\ x_2 > .5 & \text{if } x_1 > .5 \end{cases}$$

This game does not admit additional NEs in pure strategies.

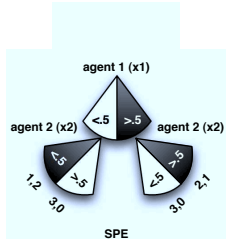


Figure 2: A representation of the unique SPE of the game reported in Fig. 1. The black slices represent the agents’ optimal strategies.

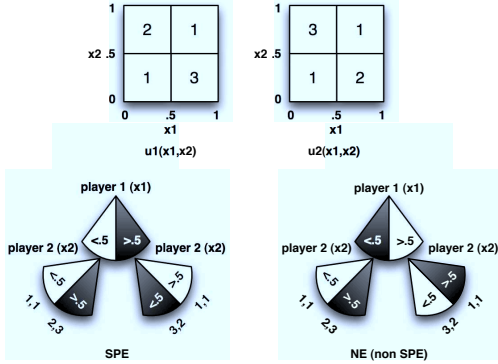


Figure 3: Agents’ utility functions and equilibrium strategies (an SPE and an NE that is not an SPE). The black slices represent the agents’ optimal strategies.

EXAMPLE 2.4. Consider a variation of the game in Fig. 1, where utilities are reported in Fig. 3. There is ‘essentially’ a unique SPE and an additional NE in pure strategies. In the (non-SPE) NE, agent 2 makes a non-credible threat, committing to play $x_2 = z$ with $z > .5$ when agent 1 plays $x_1 = w$ with $w > 0.5$, to force agent 1 to play $x_1 = w$ with $w \leq 0.5$.

Since our aim is to approximate equilibrium strategies in simulation-based games, we resort to the concepts of approximate equilibrium. Several concepts are available in the literature: ϵ -close, ϵ -Nash, and ϵ -perfect equilibria. An ϵ -close equilibrium is a strategy profile σ in which the distance (according to some metric) between every σ_i and σ_i^* is smaller than ϵ , where σ_i^* is the optimal strategy of i in an NE σ^* . This concept provides a measure of the approximation degree of the strategy. However, it is usually considered non-satisfactory and it is preferred the provision of the approximation degree of the expected utility. An ϵ -Nash equilibrium is a strategy profile σ where no agent can improve more than ϵ her utility by a unilateral deviation, while an ϵ -perfect equilibrium takes into account also the deviations of the opponents at all the subgames.

3. ALGORITHMS

The approximation of an equilibrium in simulation-based extensive-form games cannot be tackled with the algorithms proposed for strategic-form games. To use these algorithms, we need to represent the agents’ strategies as plans of actions and this is impractical.¹ This pushes for the development of *ad-hoc* algorithms that work directly on the game tree.

¹Notice that the game cannot be solved by using the algorithm described [21] assuming that agent 1 and agent 2 assign values to x_1 and x_2 respectively. This would neglect the game sequential structure.

Algorithm 1 SPE_APPROXIMATION(v)

```

1: if  $v$  is terminal then
2:   return  $\mathcal{O}(v)$ 
3: else
4:    $\{\sigma^k(v)\} \leftarrow \text{SAMPLE\_INITIALIZATION}(\rho(v))$ 
5:   while true do
6:     for each  $\sigma^k(v)$ , assign  $u^k = \text{SPE\_APPROXIMATION}(\chi(v, \sigma^k(v)))$ 
7:     if TERMINATION_CONDITION( $\{\sigma^k(v)\}, \{u^k\}$ ) then
8:       return  $\arg \max_{u \in \{u^k\}} u_{i(v)}$ 
9:     else
10:       $\{\sigma^k(v)\} \leftarrow \text{SAMPLE\_GENERATION}(\rho(v), \{\sigma^k(v)\}, \{u^k\})$ 
11:    end if
12:  end while
13: end if

```

We provide two algorithms to compute an approximate NE and an approximate SPE respectively. For sake of exposition, at first we present the algorithm for finding an approximate SPE. The algorithm for finding an approximate NE is an extension of the previous one. In both algorithms, we use $\sigma(v)$ to specify the agent $i(v)$ ’s strategy at node v .

3.1 Computing an approximate SPE

Formally, for each node $v \in V$ an SPE is defined as follows:

$$\sigma^{SPE}(v) = \arg \max_{\sigma \in \rho(v)} u_{i(v)}^{SPE}(\chi(v, \sigma)), \quad (1)$$

where $u^{SPE}(v)$ is defined as

$$u^{SPE}(v) = \begin{cases} \mathcal{O}(v) & \text{if } v \text{ is terminal} \\ u^{SPE}(\chi(v, \sigma^{SPE}(v))) & \text{otherwise.} \end{cases}$$

Algorithm 1 reports the pseudo-code to compute an approximate SPE. The algorithm, that works recursively, receives, as input, the current node v . Initially, the algorithm is called with v as the root node. If the current node v is terminal, then the oracle is called and a sample of agents’ utilities are returned (Line 2). Otherwise, the optimal strategy at v is searched as follows. Initially, one or more action samples are generated (Line 4). Then, each sample $\sigma^k(v)$ is evaluated by recursively calling SPE_APPROXIMATION on the node reachable by application of action $\sigma^k(v)$ to node v (Line 6). Given the evaluation of all the samples, if a termination condition holds (Line 7), the largest utility among those of all the samples is returned (Line 8). Otherwise, new samples are generated (Line 11). Functions SAMPLE_INITIALIZATION, SAMPLE_GENERATION, and TERMINATION_CONDITION are based on black-box non-linear optimization techniques (their description is provided in Section 4).

EXAMPLE 3.1. Consider the game in Fig. 1. Algorithm 1 works as follows. At first some samples of x_1 are generated, e.g., $\{.2, .4, .8\}$. For each sample, the algorithm is recursively called and samples for x_2 are produced, e.g., $\{.3, .5, .8\}$ for $x_1 = .2$. Then, an iterative optimization process based on resampling is carried on to optimize the value of x_2 (i.e., $\leq .5$ for $x_1 = .2$) and, finally, to return the evaluation of utility associated with the sample of $x_1 = .2$ (i.e., $u_1 = 1$). Once utility associated with all the generated samples of x_1 are evaluated, an iterative optimization process based on resampling is carried on to optimize the value of x_1 (i.e., $u_1 = 2$).

Provided that the black-box non-linear optimization technique converges to the global optimum (see Section 4.4), it is easy to show that the proposed algorithm computes the solution of problem in Equation 1.

Algorithm 2 NE_APPROXIMATION (v)

```
1: if  $v$  is non-preterminal then
2:    $\sigma(v) \leftarrow$  a random value uniformly distributed in  $\rho(v)$ 
3:    $u \leftarrow$  NE_APPROXIMATION( $\chi(v, \sigma(v))$ )
4:   while true do
5:     [ $\hat{u}, \hat{\sigma}(v)$ ]  $\leftarrow$  NE_VALIDATION( $v, v$ )
6:     if  $\hat{u}_{i(v)} \leq u(v)$  then
7:       return  $u$ 
8:     else
9:        $\sigma(v) \leftarrow \hat{\sigma}(v)$ 
10:       $u \leftarrow$  NE_APPROXIMATION( $\chi(v, \sigma(v))$ )
11:    end if
12:  end while
13: else
14:    $\{\sigma^k(v)\} \leftarrow$  SAMPLE_INITIALIZATION( $\rho(v)$ )
15:   while true do
16:     for each  $\sigma^k(v)$ , assign  $u^k = \mathcal{O}(\chi(v, \sigma^k(v)))$ 
17:     if TERMINATION_CONDITION( $\{\sigma^k(v)\}, \{u^k\}$ ) then
18:       return  $\arg \max_{u \in \{u^k\}} u_{i(v)}$ 
19:     else
20:        $\{\sigma^k(v)\} \leftarrow$  SAMPLES_GENERATION( $\rho(v), \{\sigma^k(v)\}, \{u^k\}$ )
21:     end if
22:   end while
23: end if
```

3.2 Computing an approximate NE

Surprisingly, although the NE concept poses constraints less hard than the SPE concept, the algorithm computing an NE results to be more twisted than the one computing an SPE. An NE can be formulated as in Equation 1 as far as the nodes on the equilibrium path are considered, while off the equilibrium path other agents may be non-maximizers.

Algorithm 2 depicts the pseudo-code computing an approximate NE. In the computation of an NE, two phases can be recognized. During the first phase (entirely executed by Algorithm 2), a strategy profile σ , limited to the equilibrium path, is searched. During the second phase (mainly executed by Algorithm 3), a possible strategy profile off the equilibrium path is searched such that σ is an NE. Essentially, the second phase validates that σ is an NE. The two phases iteratively alternate during the execution until an approximate NE has not been found. The details follow.

(First phase) In Algorithm 2, for each non-preterminal node² v a random strategy is assigned to $\sigma(v)$ (Line 2) and the algorithm is recursively called on the next node reached by applying $\sigma(v)$ to the current node (Line 3). If v is preterminal, the best strategy of agent $i(v)$ at v is searched by using black-box optimization techniques (Lines 14–22). Notice that this optimization works exactly as Lines 6–13 of Algorithm 1 except that here the evaluation of the sample is directly accomplished by calling the oracle without any recursive call of the algorithm. Once Line 18 is executed, a complete strategy assignment σ from the root of the tree to a terminal node (along a single path) is built.

EXAMPLE 3.2. Consider the game depicted in Fig. 3. Algorithm 2 randomly assigns a value to x_1 , e.g., $x_1 = 0.3$, and subsequently the optimal value of x_2 is found, i.e., $x_2 \geq .5$.

(Second phase) The algorithm tries to validate that the found strategy profile σ is an NE (Line 5). This is accomplished by Algorithm 3. Given a node v , Algorithm 3 searches for a strategy in the subgames such that agent $i(v)$ cannot gain more by deviating from her strategy prescribed

²A node v is preterminal if, once applied an action to v , a terminal node is reached.

Algorithm 3 NE_VALIDATION (v, v_0)

```
1: if  $v$  is terminal then
2:   return  $\mathcal{O}(v)$ 
3: else
4:    $\{\sigma^k(v)\} \leftarrow$  SAMPLE_INITIALIZATION( $\rho(v)$ )
5:   while true do
6:     for each  $\sigma^k(v)$ , assign  $u^k =$  NE_VALIDATION( $\chi(v, \sigma^k(v)), v_0$ )
7:     if  $i(v) = i(v_0)$  then
8:       if TERMINATION_CONDITION( $\{\sigma^k(v)\}, \{u^k_{i(v)}\}$ ) then
9:         return [ $\arg \max_{u \in \{u^k\}} u_{i(v_0)}$ , best  $\sigma^k(v)$ ]
10:      else
11:         $\{\sigma^k(v)\} \leftarrow$  SAMPLES_GENERATION( $\rho(v), \{\sigma^k(v)\}, \{u^k_{i(v)}\}$ )
12:      end if
13:    else
14:      if TERMINATION_CONDITION( $\{\sigma^k(v)\}, \{-u^k_{i(v_0)}\}$ ) then
15:        return [ $\arg \min_{u \in \{u^k\}} u_{i(v_0)}$ , best  $\sigma^k(v)$ ]
16:      else
17:         $\{\sigma^k(v)\} \leftarrow$  SAMPLES_GENERATION( $\rho(v), \{\sigma^k(v)\}, \{-u^k_{i(v_0)}\}$ )
18:      end if
19:    end if
20:  end while
21: end if
```

by σ . Differently from what happens in the case of SPE, the strategy in the subgames of v does not need to be sequentially rational. Practically, this means that, while agent $i(v)$ will maximize her utility in such subgames, the behavior of her opponents is free, they do not necessarily maximize their utility. To assure that there is not any strategy such that $i(v)$ can gain more by deviating from her strategy prescribed by σ at v , we assume that all the opponents of $i(v)$ behave in the attempt to minimize the utility of $i(v)$. In this way, we find the maxmin value of agent $i(v)$ from the subgames. If the maxmin value is larger than the utility given by the strategy prescribed by σ , then σ is not an NE, otherwise there is at least a strategy of the subgames such that the strategy prescribed by σ is optimal. The validation process must be repeated at every node v . If a strategy is not validated at a given node v , Algorithm 2 assigns the maxmin strategy of the subgames to $\sigma(v)$ (Line 8) and phase 1 is restarted.

Algorithm 3 works similarly to Algorithm 1 except that the opponents of agent $i(v_0)$, where v_0 is the node whose strategy we are validating, minimize agent $i(v_0)$'s utility (Line 15). We use the same black-box optimization techniques used in the previous algorithms, passing $\{-u^k_{i(v_0)}\}$ when we need to minimize (Lines 14 and 17).

EXAMPLE 3.3. Consider the game depicted in Fig. 3. Suppose that Algorithm 2 has assigned $x_1 = .3$ and $x_2 = .1$, and that NE_VALIDATION is executed to validate the strategy of agent 1. A number of samples are generated for x_1 . For each sample of x_1 , the optimal value of x_2 minimizing agent 1's utility is found. It can be easily observed that the maxmin utility of agent 1 from all the subgames is 1 and therefore the initial strategy is validated.

The efficiency of Algorithm 3 can be improved by using a pruning procedure similar to the alpha-beta pruning. More precisely, call v_0 the node whose strategy is to validate. To assert that a given strategy σ is not an NE, we do not need to compute the maxmin value of v_0 , but it is enough to find a strategy $\hat{\sigma}_{i(v_0)}$ such that agent $i(v_0)$'s utility is larger than

the one provided by the strategy to validate. Moreover, in all the subgames of v_0 , we do not strictly need that opponents of agent $\iota(v_0)$ find the minimum of agent $\iota(v_0)$'s utility, but it is sufficient that they find an action such that agent $\iota(v_0)$'s utility is not larger than the one provided by the strategy to validate. Therefore, the termination condition at Line 8 can be safely modified in the following way: as soon as a sample provides agent $\iota(v_0)$ with a utility larger than the one provided by the strategy to be validated, the algorithm returns such utility and the corresponding sample. Similarly, at Line 14 it is possible to make the algorithm return as soon as a sample, that provides agent $\iota(v_0)$ with a utility non-larger than the one provided by the strategy to be validated, is found.

EXAMPLE 3.4. *Consider Example 3.3. Every time a sample of x_2 is evaluated and it provides a utility of 1 to agent 1, no further samples are generated.*

4. OPTIMIZATION ALGORITHMS

As shown in the previous section, the computation of an SPE and an NE in extensive-form games requires to solve a number of recursive optimization problems. For each node v we need to maximize an unknown objective function which depends on the solutions of other optimization problems defined in the nodes of the sub-tree rooted at v . In a continuous extensive-form game, this means to solve a number of unconstrained continuous optimization problems. In the literature, researchers have proposed many optimization methods, which can be split into two main categories: deterministic methods (e.g., real algebraic geometry [2], Lipschitz optimization [18]) and non-deterministic (or stochastic) ones (e.g., evolutionary algorithms [7], simulated annealing [10], cross-entropy method [16], particle swarm optimization [9]). The optimization process which is common to all these algorithms is synthetically summarized in Algorithm 4. Starting from one or more initial samples within the search space D , at each iteration, new candidate solutions are generated and, on the basis of their scores (i.e., the corresponding objective function values) and eventually other information about the objective function, the search is directed towards the most promising regions in the search space. The search process iterates until some termination condition is met: for instance, many optimization algorithms stop when the improvement in a sequence of consecutive iterations falls below a predefined threshold or a predefined maximum number of iterations is reached. In the following subsections we will focus on one deterministic algorithm (Lipschitz optimization) and two random-search approaches (simulated annealing and cross-entropy method), and we will describe how the main steps of the search process are implemented in each of them. For sake of simplicity, we will present the algorithms for one-dimensional optimizations, but extensions to multiple dimensions are straightforward [1].

4.1 Lipschitz optimization

Lipschitz optimization is a deterministic approach to the global optimization problem. The basic assumption of this approach is that the objective function $u(x)$ satisfies the Lipschitz condition over the closed optimization interval $[a, b]$. A function is Lipschitz if there exists a finite bound α (called Lipschitz constant) to its rate of change:

$$|u(x_1) - u(x_2)| \leq \alpha |x_2 - x_1|, \quad x_1, x_2 \in [a, b]. \quad (2)$$

Algorithm 4 Optimization Algorithm

```

1:  $i := 0$ 
2:  $\{x_0^k\}_{1 \leq k \leq N} = \text{SAMPLE\_INITIALIZATION}(D)$ 
3: repeat
4:    $i := i + 1$ 
5:    $\{x_i^k\}_{1 \leq k \leq N} = \text{SAMPLE\_G}(D, \{x_{[0:i-1]}^k\}_{1 \leq k \leq N}, \{y_{[0:i-1]}^k\}_{1 \leq k \leq N})$ 
6:   for  $j = 1$  to  $N$ , assign  $y_i^j = u(x_i^j)$ 
7: until  $\text{TERMINATION\_CONDITION}(\{x_{[0:i]}^k\}_{1 \leq k \leq N}, \{y_{[0:i]}^k\}_{1 \leq k \leq N})$ 
8: return  $\arg \max_{x \in \{x_i^k\}_{1 \leq k \leq N}} u(x)$ 

```

The key idea of Lipschitz optimization is to select the next query point by maximizing an upper-bound function which is built on the sequence of samples generated by the search process.

Sample initialization. The algorithm starts by generating two samples placed at the boundary of the optimization interval $[a, b]$: $\{x_0^k\} = \{a, b\}$, and computes the corresponding scores $u(a)$ and $u(b)$.

Sample generation. In the following iterations, to select the next sample of the search process, the algorithm, given all the previously generated samples $\{x_{[0:i-1]}^k\}$ (sorted by value) and the associated scores $\{u(x_{[0:i-1]}^k)\}$, determines the interval $[x^j, x^{j+1}]$ containing the highest value of the upper-bound function:

$$[x^{j*}, x^{j+1*}] = \arg \max_{x^j, x^{j+1} \in \{x_{[0:i-1]}^k\}} \frac{u(x^j) + u(x^{j+1})}{2} + \alpha \cdot \frac{x^{j+1} - x^j}{2}.$$

Once the interval has been identified, the new sample will be generated in the following point:

$$x_i = \frac{u(x^{j+1*}) - u(x^{j*})}{2\alpha} + \frac{x^{j*} + x^{j+1*}}{2}.$$

Termination condition. The stop criterion for Lipschitz optimization is that the difference between the actual objective-function value and the upper-bound value is lower than a specified global tolerance ϵ_{stop} .

4.2 Simulated Annealing

Simulated Annealing (SA) is a well-known probabilistic metaheuristics algorithm for finding global maximum (or minimum) of an objective function. It works by emulating the physics process whereby a solid is at first heated and then slowly cooled to find a configuration with lower internal energy than the initial one. The idea behind SA is to take a random walk through the search space at successively lower temperatures (i.e., less exploration), where the probability of taking a step is given by a Boltzmann distribution. Although SA is often used when the search space is discrete, here we consider its application to continuous global optimization problems [12].

Sample initialization. The SA algorithm starts from a random sample drawn from a uniform distribution over the search space.

Sample generation. At each iteration i , a new candidate sample \hat{x} is generated from a kernel distribution $\mathcal{K}(\cdot, x_{i-1})$ (whose definition is critical to the convergence of SA [6]) centered in the last available sample x_{i-1} (we used Gaussian kernels). If the new sample \hat{x} has a score higher than the one of x_{i-1} the sample is accepted. Otherwise, to avoid getting trapped in local maxima, it can be still accepted with

a likelihood that is proportional to the temperature parameter τ and the score difference $u(\hat{x}) - u(x_{i-1})$ (Metropolis algorithm [15]):

$$x_i = \begin{cases} \hat{x} & \text{if } p \leq \min \left\{ 1, e^{-\frac{u(\hat{x}) - u(x_{i-1})}{\tau}} \right\} \\ x_{i-1} & \text{otherwise,} \end{cases}$$

where p is a random number drawn uniformly in $[0, 1]$. As the search process goes on, in order to converge to a solution the temperature parameter should decrease according to some cooling scheme [3].

Termination condition. Usually, SA algorithms stop when new samples are consecutively rejected for a fixed number of iterations. In this paper, we use a different criterion (similar to the one adopted in [19]) based on the score of the samples considered in the last M iterations. In particular, the algorithm is stopped at the i -th iteration if:

$$\max_{j \in [i-M, i]} u(x_j) - \frac{1}{M} \sum_{j=i-M}^i u(x_j) < \epsilon_{STOP}.$$

In words, the algorithm is stopped when its progress is considered too small.

4.3 Cross-entropy method

The Cross-Entropy (CE) method is a Monte Carlo technique to solve optimization problems. The method consists of two main steps: 1) generate samples according to a probability density defined over the search space, 2) update the probability density parameters by minimizing the cross-entropy (i.e., Kullback-Liebert divergence [11]) with respect to the best samples (*elite* samples). In this way, new samples will be generated in the most promising regions of the search space. Since considering distributions from an exponential family such minimization can be solved analytically, in this paper we use Gaussian densities parameterized by the mean μ and variance σ^2 .

Sample initialization. At the first iteration, since no prior information is available, k samples are randomly generated using a uniform distribution over the search space.

Sample generation. At each iteration i , given the sample scores of the previous iteration $\{u(x_{i-1}^k)\}_{1 \leq k \leq N}$, a percentage ρ_{CE} of the best samples is selected to estimate the new probability density from which new samples will be drawn in the next iteration. Using Gaussian densities, the minimization of the cross-entropy measure leads to update the parameters of the distribution by simply computing the sample mean and sample variance of the elite samples, i.e., the best $\lceil N \cdot \rho_{CE} \rceil$ samples.

Termination condition. The algorithm is stopped when the improvement in the $(1 - \rho_{CE})$ -quantile (i.e., the score of the worst elite sample) does not exceed ϵ_{STOP} for d_{CE} successive iterations, or when the maximum number of iterations t_{max} is reached.

4.4 Convergence Properties

Lipschitz optimization allows one to approximate with arbitrary precision the global maximum of Lipschitz functions when an upper bound to the function derivative (the Lipschitz constant) is known. Since Lipschitz optimization is a deterministic method with very few parameters, there is no need for multiple runs and parameter tuning is minimized. On the other hand, when the Lipschitz constant is unknown or the objective function is not Lipschitz, no guarantees of accuracy can be given. Furthermore, when objective func-

tions have large Lipschitz constants and/or are defined over multiple dimensions, the convergence is very slow.

When no information about the objective function is available, randomized-search methods (e.g., simulated annealing and cross-entropy method) are usually considered. Both the algorithms described in this section are simple and effective approaches to solve continuous optimization problems, even if simulated annealing is a local search algorithm (whose performance depends critically on a proper choice of the cooling scheme), while the cross-entropy method is a global optimization method. It is possible to show that, under quite mild conditions on the kernel distributions and the objective function, such methods converge to the optimum with probability approaching one as the number of samples grows to infinity (for details refer to [6, 13]).

5. EXPERIMENTAL EVALUATION

We implemented our algorithms with Matlab R10 and we executed them with a UNIX computer with dual quad-core 2.33GHz CPU and 8GB RAM. Our experimental activity is structured as follows. Initially, we apply our algorithms to a well-known practical economic problem (i.e., bargaining) modeled as a continuous extensive-form game which presents piecewise linear utility functions and whose exact solution is known in closed form. Subsequently, we apply our algorithms to *ad-hoc* games with a class of highly non-linear utility functions widely studied in multi-agent systems.

5.1 The bargaining case study

We chose the alternating-offers game as case study, being the principal model for strategic bargaining. The alternating-offers game prescribes that two agents, a buyer \mathbf{b} and a seller \mathbf{s} , play alternately at discrete time points. Time t is discrete and $\iota(t)$ is defined as follows: $\iota(0)$ is a parameter of the problem and for $t > 0$ it is such that $\iota(t) \neq \iota(t-1)$. The pure strategies available to agent $\iota(t)$ at $t > 0$ are: *offer*(\bar{x}), where $\bar{x} \in [0, 1]$; *accept*, that concludes the game with outcome (\bar{x}, t) , where \bar{x} is the value offered at $t-1$, and t is the time point at which the offer is accepted; and *exit*, that concludes the game with outcome *NoAgreement*. At $t=0$ only actions *offer*(\bar{x}) and *exit* are available. Agents' utility functions are defined as follows. Each agent i has a deadline T_i . Before the deadlines, utility functions are defined as $U_{\mathbf{b}}(x, t) = (1-x) \cdot (\delta_{\mathbf{b}})^t$ for the buyer and $U_{\mathbf{s}}(x, t) = x \cdot (\delta_{\mathbf{s}})^t$ for the seller. After the deadline of agent i , her utility is -1 . δ_i and T_i are parameters. The alternating-offers game admits a unique SPE that prescribes that at each time t there is an optimal offer $x^*(t)$ for every $t \leq \min\{T_{\mathbf{b}}, T_{\mathbf{s}}\}$ such that agent $\iota(t)$ makes it and her opponent accepts it at $t+1$. (For the computation of x^* we point the interested reader to [4]) There is no optimal offer at $t > \min\{T_{\mathbf{b}}, T_{\mathbf{s}}\}$, so agents' optimal action is *exit*. That is, the minimal deadline essentially defines the depth of the game tree (exactly, the tree depth is $\min\{T_{\mathbf{b}}, T_{\mathbf{s}}\} + 1$).

We generated some game instances with different minimal deadlines from the range $\{3, 4, 5, 6\}$. We developed simple variations of our algorithms to capture the fact that actions are mixed, combining discrete and continuous actions. Moreover, we force agents to exit when the minimal deadline is expired (otherwise agents can indefinitely play).

At first, we have evaluated the performance of Algorithm 1 with different parameterizations when the minimal deadlines is 3. The used parameterizations and the obtained results

s.p.i.	best samples per iteration						
	2		4		6		
	$E[\cdot]$	std	$E[\cdot]$	std	$E[\cdot]$	std	
10	0.078	0.084	0.082	0.157	0.195	0.186	ϵ
	$3 \cdot 10^3$	$1 \cdot 10^4$	$3 \cdot 10^4$	$9 \cdot 10^3$	$5 \cdot 10^4$	$3 \cdot 10^4$	e.s.
	0.763	0.329	1.684	0.613	2.111	0.864	time (s)
20	0.042	0.045	0.015	0.009	0.017	0.013	ϵ
	$5 \cdot 10^4$	$2 \cdot 10^4$	$2 \cdot 10^5$	$3 \cdot 10^4$	$3 \cdot 10^5$	$7 \cdot 10^4$	e.s.
	2.677	0.730	5.725	0.836	7.987	2.295	time (s)
30	0.036	0.038	0.0048	0.003	0.010	0.003	ϵ
	$4 \cdot 10^5$	$9 \cdot 10^4$	$6 \cdot 10^5$	$9 \cdot 10^4$	$8 \cdot 10^5$	$6 \cdot 10^4$	e.s.
	10.431	2.290	13.991	1.811	20.511	2.038	time (s)
40	0.024	0.016	0.015	0.025	0.005	0.007	ϵ
	$9 \cdot 10^5$	$1 \cdot 10^5$	$1 \cdot 10^6$	$2 \cdot 10^5$	$1 \cdot 10^6$	$2 \cdot 10^5$	e.s.
	18.983	3.789	31.244	7.848	36.310	6.102	time (s)
50	0.013	0.009	0.009	0.008	0.003	0.003	ϵ
	$2 \cdot 10^6$	$3 \cdot 10^5$	$3 \cdot 10^6$	$4 \cdot 10^5$	$3 \cdot 10^6$	$4 \cdot 10^5$	e.s.
	32.801	6.005	49.719	7.551	54.853	7.412	time (s)

Table 1: Experimental results with a bargaining game with depth 3 obtained by applying CE to compute SPE (‘s.p.i.’ means samples per iteration and ‘e.s.’ means evaluated samples).

M	temperature (τ)						
	0.3		0.5		0.7		
	$E[\cdot]$	std	$E[\cdot]$	std	$E[\cdot]$	std	
10	0.042	0.030	0.082	0.133	0.031	0.033	ϵ
	$3 \cdot 10^4$	$1 \cdot 10^5$	$4 \cdot 10^4$	$1 \cdot 10^5$	$4 \cdot 10^4$	$2 \cdot 10^5$	e.s.
	0.915	0.051	0.946	0.024	0.956	0.055	time (s)
20	0.027	0.019	0.026	0.024	0.026	0.018	ϵ
	$3 \cdot 10^5$	$3 \cdot 10^5$	$3 \cdot 10^5$	$9 \cdot 10^4$	$3 \cdot 10^5$	$1 \cdot 10^4$	e.s.
	6.741	0.098	6.411	0.353	6.482	0.252	time (s)
30	0.023	0.017	0.020	0.017	0.029	0.024	ϵ
	$1 \cdot 10^6$	$3 \cdot 10^4$	$1 \cdot 10^6$	$3 \cdot 10^4$	$1 \cdot 10^6$	$3 \cdot 10^4$	e.s.
	22.650	0.782	22.820	0.910	24.390	0.623	time (s)
40	0.018	0.014	0.022	0.021	0.019	0.019	ϵ
	$2 \cdot 10^6$	$5 \cdot 10^4$	$2 \cdot 10^6$	$7 \cdot 10^4$	$2 \cdot 10^6$	$7 \cdot 10^4$	e.s.
	54.590	1.154	55.152	1.739	54.322	1.426	time (s)
50	0.018	0.017	0.013	0.181	0.035	0.062	ϵ
	$5 \cdot 10^6$	$1 \cdot 10^5$	$5 \cdot 10^6$	$8 \cdot 10^4$	$5 \cdot 10^6$	$1 \cdot 10^5$	e.s.
	108.627	1.903	107.689	2.865	106.613	2.781	time (s)

Table 2: Experimental results with a bargaining game with depth 3 obtained by applying SA to compute SPE (‘M’ is defined in Section 4.2 and ‘e.s.’ means evaluated samples).

related to CE optimization are reported in Tab. 1, those related to SA optimization are reported in Tab. 2, and those related to Lipschitz optimization are reported in Tab. 3 (notice that the utilities in the bargaining game are no Lipschitz, not being continuous). We evaluated our algorithm in terms of ϵ value of the approximate ϵ -perfect equilibrium found by the algorithms, number of evaluated samples (e.s.), and computational time. The results reported in the tables are averaged over 10 executions ($\epsilon_{STOP} = 10^{-2}$ for CE and SA).

CE and SA exhibit similar performance in terms of ϵ value. This value reduces when the number of samples per iteration (s.p.i.) and M increase, while there are not optimal values of elite samples and temperature independently of the number of samples per generation. Only with a Lipschitz constant larger than 3, Lipschitz optimization returns a value of ϵ comparable to that returned by the other two optimization techniques. On the other hand, CE and SA evaluated a strictly smaller number of samples (ϵ being equal, CE always outperforms SA). Finally, computational times are proportional to the number of e.s. for all the optimization techniques in the same way (for this reason, we omit the computational time in the following evaluations).

Tab. 4 and Tab. 5 report the performance of Algorithm 1 with CE and SA with their fastest configurations (10 s.p.i. and 2 elite samples for CE, and $M = 2$ and $\tau = 0.3$ for SA) for different values of the minimal deadline. The results are

Lipschitz constant (α)				
1	2	3	4	
0.221	0.143	0.073	–	ϵ
138,359	1,127,485	6,692,909	$> 10^7$	e.s.
10.85	108.52	577.221	–	time (s)

Table 3: Experimental results when computing SPE with Lipschitz optimization in a bargaining game with depth 3 (‘e.s.’ means evaluated samples).

minimal deadline	samples per iteration					
	10		15		20	
	ϵ	e.s.	ϵ	e.s.	ϵ	e.s.
3	0.078	33,017	0.065	41,674	0.042	48,259
4	0.056	954,356	0.033	4,135,410	0.031	6,259,260
5	0.043	14,354,760	0.039	50,714,660	0.035	73,801,294
6	0.051	270,564,843	–	$> 10^9$	–	$> 10^9$

Table 4: Experimental results when computing SPE in a bargaining game using CE with elite samples equal to 2 (‘e.s.’ means evaluated samples).

averaged over 10 executions. We observe that the ϵ value is rather small even with minimal deadline equal to 6. The number of e.s. rises exponentially in the length of the minimal deadline. Also in this case CE outperforms SA in terms of evaluated samples.

We evaluate Algorithm 2 with the parameterizations used in Tab. 4 for CE. We report only e.s. because the ϵ -Nash value is zero for all the executions. Finding an NE requires less samples than finding an SPE, thus allowing one to approximate a bargaining problem with deadline 20.

5.2 Games with rugged utility functions

We evaluate the performance of our algorithms when utility functions are highly non-linear. Non-linear utility functions are deeply studied in the negotiation field and a class of utility functions that has received a lot of attention is said *rugged utility* [14]. A rugged utility function is defined as follows. Call (x_1, \dots, x_n) the arguments of utility function U where $x_i \in [0, 1]$. Each domain $[0, 1]$ is divided into k intervals where the j -th interval is $[\frac{j-1}{k}, \frac{j}{k}]$. Call $int : [0, 1] \rightarrow \{1, \dots, k\}$ the function that, given a continuous value x_i , returns the interval to which x_i belongs. Utility U is defined as $U(x_1, \dots, x_n) = RAND \frac{1}{n} \sum_i^n int(x_i)$ where $RAND$ is a number randomly drawn from a uniform probability distribution over $[0, 1]$. With these utility functions SPEs can be computed exactly.

We generated game trees with a depth $\in \{3, 4, 5\}$ and rugged utility functions. We executed 10 times Algorithm 1 with cross entropy for each configuration reported in Tab 7 (the number of elite samples is equal to 2). We compared the results returned by our algorithms with respect to the SPE of the game. We report in Tab. 7 the results with tree depth equal to 3 (the results with 4 and 5 are similar, but they require a much larger number of e.s.): the success percentage (suc.), the ϵ value of the associated ϵ -approximate equilibrium, and the number of evaluated samples.

It can be observed that the results with rugged utility functions are worse than those obtained in the bargaining case study. More precisely, the ϵ value and the number of e.s. are larger than those obtained in Section 5.1. The performance decreases with the increasing of k . In order to have a satisfactory success percentage, the number of s.p.i. must be rather larger than k . This poses severe limits to the size of the game trees solvable by the algorithm within reasonable time. We applied Algorithm 2 with CE using the same parameterization and with minimal deadline larger than 3. As

minimal deadline	samples per iteration					
	10		15		20	
	ϵ	e.s.	ϵ	e.s.	ϵ	e.s.
3	0.078	33,017	0.065	41,674	0.042	48,259
4	0.055	1,950,350	0.062	12,295,395	0.057	43,368,477
5	0.047	89,769,110	–	$> 10^9$	–	$> 10^9$
6	–	$> 10^9$	–	$> 10^9$	–	$> 10^9$

Table 5: Experimental results when computing SPE in a bargaining game using SA with $\tau = 0.3$.

minimal deadline	samples per iteration		
	10	15	20
3	532	874	1,146
4	1,361	1,983	3,214
5	2,841	3,324	5,482
6	5,362	8,641	9,215
10	60,316	93,325	341,513
15	301,142	762,413	1,356,234
20	1,882,582	6,241,562	13,646,221

Table 6: Evaluated samples when computing an NE in a bargaining game using CE with elite samples equal to 2 .

in the bargaining case, the computation of an NE requires a strictly smaller number of evaluated samples: (with 2 elite samples) 10^3 e.s. with deadlines 3, 10^4 e.s. with deadlines ≤ 7 , 10^5 e.s. with deadlines ≤ 12 .

6. CONCLUSIONS

Simulation-based games (i.e., games in which the agents’ payoffs are provided as the result of a simulation process) have received a lot of attention in the scientific community. In this paper, we extended such class of games when the games are in extensive form and have continuous actions. We provided two convergent algorithms to compute an approximate subgame perfect and an approximate Nash equilibrium respectively. We used different black-box optimization techniques (simulated annealing, cross entropy, and Lipschitz optimization) in our algorithms and we experimentally evaluated them with different settings. A subgame perfect equilibrium can be computed in game trees with a small depth, while the computation of a Nash equilibrium is easier. Furthermore, the number of evaluated samples being equal, cross-entropy optimization demonstrated to outperform simulate annealing and Lipschitz optimization.

In future works, we try to improve the efficiency of our algorithms for all the situations wherein information on the structure of the problem is available, e.g., when games have an action-graphical structure.

7. REFERENCES

- [1] H. Benson, R. Horst, and P. Pardalos. Handbook of Global Optimization, 1995.
- [2] J. Bochnak, M. Coste, and M. Roy. *Real algebraic geometry*. Springer Verlag, 1998.
- [3] I. Bohachevsky, M. Johnson, and M. Stein. Generalized simulated annealing for function optimization. *TECHNOMETRICS*, 28(3):209–217, 1986.
- [4] F. Di Giunta and N. Gatti. Bargaining over multiple issues in finite horizon alternating-offers protocol. *ANN MATH ARTIF INTEL*, 47(3-4):251–271, 2006.
- [5] D. Fudenberg and J. Tirole. *Game Theory*. The MIT Press, Cambridge, USA, 1991.
- [6] A. Ghate and R. Smith. Adaptive search with stochastic acceptance probabilities for global

k	samples per iteration								
	10			20			30		
		ϵ	e.s.		ϵ	e.s.		ϵ	e.s.
10	40%	0.060	$2 \cdot 10^4$	80%	0.036	$2 \cdot 10^5$	90%	0.022	$1 \cdot 10^6$
20	10%	0.074	$2 \cdot 10^4$	30%	0.069	$2 \cdot 10^5$	50%	0.028	$1 \cdot 10^6$
30	0%	0.083	$3 \cdot 10^4$	10%	0.077	$3 \cdot 10^5$	20%	0.052	$1 \cdot 10^6$
40	0%	0.098	$3 \cdot 10^4$	0%	0.082	$3 \cdot 10^5$	10%	0.060	$1 \cdot 10^6$
50	0%	0.153	$4 \cdot 10^4$	0%	0.089	$3 \cdot 10^5$	0%	0.064	$1 \cdot 10^6$
100	0%	0.234	$6 \cdot 10^4$	0%	0.156	$5 \cdot 10^5$	0%	0.145	$1 \cdot 10^6$
	suc.	ϵ	e.s.	suc.	ϵ	e.s.	suc.	ϵ	e.s.

Table 7: Experimental results when computing SPE with rugged function using CE (‘e.s.’ means evaluated samples and ‘suc.’ means success probability).

- optimization. *OPER RES LETT*, 36(3):285–290, 2008.
- [7] D. Goldberg. *Genetic algorithms in search, optimization, and machine learning*. Addison-Wesley, 1989.
- [8] C. Harris. Existence and characterization of perfect equilibrium in games of perfect information. *ECONOMETRICA*, 53:613–628, 1985.
- [9] J. Kennedy and R. Eberhart. Particle swarm optimization. In *ICNN*, volume 4, pages 1942–1948, 1995.
- [10] S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. *SCIENCE*, 220(4598):671–680, 1983.
- [11] S. Kullback and R. Leibler. On information and sufficiency. *ANN MATH STAT*, pages 79–86, 1951.
- [12] M. Locatelli. Simulated annealing algorithms for continuous global optimization: convergence conditions. *J OPTIMIZ THEORY APP*, 104(1):121–133, 2000.
- [13] L. Margolin. On the convergence of the cross-entropy method. *ANN OPER RES*, 134(1):201–214, 2005.
- [14] I. Marsa-Maestre, M. Lopez-Carmona., J. Velasco, and E. de la Hoz. Avoiding the prisoner’s dilemma in auction-based negotiations for highly rugged utility spaces. In *AAMAS*, pages 425–432, 2010.
- [15] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, E. Teller, et al. Equation of state calculations by fast computing machines. *J CHEM PHYS*, 21(6):1087, 1953.
- [16] R. Rubinstein and D. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*. Springer-Verlag, 2004.
- [17] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game Theoretic and Logical Foundations*. Cambridge University Press, 2008.
- [18] B. Shubert. A sequential method seeking the global maximum of a function. *SIAM J NUMER ANAL*, 9(3):379–388, 1972.
- [19] D. Vanderbilt and S. Louie. A Monte Carlo simulated annealing approach to optimization over continuous variables. *J COMPUT PHYS*, 56(2):259–271, 1984.
- [20] Y. Vorobeychik, D. Reeves, and M. Wellman. Constrained automated mechanism design for infinite games of incomplete information. In *UAI*, 2007.
- [21] Y. Vorobeychik and M. Wellman. Stochastic search methods for nash equilibrium approximation in simulation-based games. In *AAMAS*, pages 1055–1062, 2008.

Maximum Causal Entropy Correlated Equilibria for Markov Games

Brian D. Ziebart, J. Andrew Bagnell, Anind K. Dey
Carnegie Mellon University
bziebart@cs.cmu.edu, dbagnell@ri.cmu.edu, anind@cs.cmu.edu

ABSTRACT

Motivated by a machine learning perspective—that game-theoretic equilibria constraints should serve as guidelines for predicting agents’ strategies, we introduce maximum causal entropy correlated equilibria (MCECE), a novel solution concept for general-sum Markov games. In line with this perspective, a MCECE strategy profile is a uniquely-defined joint probability distribution over actions for each game state that minimizes the worst-case prediction of agents’ actions under log-loss. Equivalently, it maximizes the worst-case growth rate for gambling on the sequences of agents’ joint actions under uniform odds. We present a convex optimization technique for obtaining MCECE strategy profiles that resembles value iteration in finite-horizon games. We assess the predictive benefits of our approach by predicting the strategies generated by previously proposed correlated equilibria solution concepts, and compare against those previous approaches on that same prediction task.

Categories and Subject Descriptors

H.4 [Information Systems Applications]: Miscellaneous

General Terms

Theory, Algorithms

Keywords

Game theory, correlated equilibria, maximum entropy

1. INTRODUCTION

Agents often need to predict the future behavior of other agents [9] to appropriately choose their own actions. Equilibria solution concepts, such as Nash equilibria [22], and the more general correlated equilibria (CE) [1], which allow agents to coordinate their actions, are important constructs for multi-agent games that provide certain individual or group performance guarantees based on assumed rationality. Agents playing many decentralized, adaptive strategies (such as no-regret learning) will converge to CE [23, 10, 14, 11], but the particular set of convergence CE will vary depending on the strategies employed. From an applied machine learning perspective, existing equilibria concepts are

Cite as: Maximum Causal Entropy Correlated Equilibria for Markov Games, Brian D. Ziebart, J. Andrew Bagnell and Anind K. Dey, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 207-214.
Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

often not useful for prediction. First, they generally do not fully specify a unique strategy profile, making strategy prediction under-specified without additional assumptions. Second, they are typically not designed to provide any predictive performance guarantees.

We introduce the maximum causal entropy correlated equilibria (MCECE) solution concept to enable equilibria-based prediction for general-sum Markov games. It extends maximum entropy correlated equilibria (MaxEntCE) for normal-form games [24] to the dynamic game setting by specifying the unique CE strategy profile with the *fewest* additional assumptions. This property is useful for three main purposes. First, for prescriptive settings, the resulting MCECE strategy profile best conceals the underlying motives of agents in a manner we specify in Section 3. This can often be an important consideration when revealing too much information can lead to future exploitation. Second, for predictive purposes, the MCECE strategy profile minimizes the worst-case log-loss when predicting the actions of agents assumed to act according to an unknown CE strategy. Thus, it is theoretically justified for predicting the actions of agents assumed to be jointly behaving rationally. Third, for gambling on the sequence of agents’ actions, the MCECE strategy profile maximizes the worst-case expected investment growth rate under uniform odds.

We present in Section 4 an efficient algorithm for obtaining MCECE based on convex optimization that ultimately reduces to a dynamic programming algorithm over time steps of finite games. In contrast with our predictive approach, previously developed CE solution concepts impose very strong assumptions on agents’ preference over possible CE strategy profiles to provide unique payoffs [20, 15, 12]. In Section 5, we evaluate the predictive benefits of the MCECE and other strategy profiles at predicting the strategies of one another.

2. BACKGROUND

We first review concepts in game theory and information theory to properly situate the contributions of this paper.

2.1 Games and Equilibria

The canonical set of games studied within game theory are one-shot games with matrices of payoffs.

Definition 1. A **normal-form game**, is defined by a set of agents N , a set of joint agent actions A , and a utility vector $U : A \mapsto \mathbb{R}^N$, specifying the payoffs for each agent $i \in N$ for joint action $a \in A$. Each agent controls a portion $a_i \in A_i$ of the joint action $a = \times_{i \in N} a_i$.

In a normal-form game (*Definition 1*), each agent ($i \in N$) simultaneously selects an action ($a_i \in A_i$) and receives a numerical **payoff**, $U_{a,i} \in \mathbb{R}$, based on the combination of actions, $a \in A$.

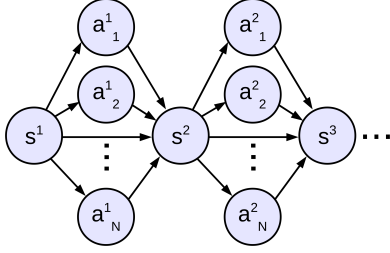


Figure 1: The sequence of states and (Markovian) actions of a Markov game.

Markov games (*Definition 2*) generalize normal-form games to sequential settings. In a Markov game, the joint actions of N agents at time t , denoted a^t , stochastically lead to a next state as shown in Figure 1.

Definition 2. A **Markov game** is defined by a set of states (S) representing the joint states of N agents, a set of actions (A), a probabilistic state transition function, $T : S \times A \mapsto \Delta_S$, and a utility function, $Utility_i : N \times S \times A \mapsto \mathbb{R}$.

Agents choose **strategy profiles**, $\pi \in \Delta_A$, specifying next actions for each situation that are either **mixed** (*i.e.*, stochastic) or **pure** (*i.e.*, deterministic); and either **correlated** (*i.e.*, joint functions) or **independent** based on a (discounted, $0 < \gamma \leq 1$) cumulative expected utility:

$$\begin{aligned} \text{ExpUtil}_i^\pi(a^t, s^t) & \quad (1) \\ \triangleq \mathbb{E}_{S^{t+1}:T, A^{t+1}:T} \left[\sum_{\tau \geq t} \gamma^\tau \text{Utility}_i(s^\tau, a^\tau) \middle| a^t, s^t, \pi \right], \end{aligned}$$

where we denote the variables being marginalized over in the expectation using subscript. We assume in Equation 1 and throughout this paper that the strategy profile is mixed and **Markovian**¹, meaning it depends only on the current state and time step.

To obtain strategy profiles, it is useful to consider the amount of utility gained by switching from a provided action, a_i^t , to an alternate action, $a_i^{t'}$, called a **deviation action**, when: all agents' actions, a^t , are known (Equation 2); or when other agents' actions, denoted $a_{-i}^t \in A_{-i}^t$, are unknown and averaged over according to the strategy profile, π (Equation 3):

$$\begin{aligned} \text{ExpDevGain}_i^\pi(a^t, s^t, a_i^{t'}) & \quad (2) \\ \triangleq \text{ExpUtil}_i^\pi(\{a_{-i}^t, a_i^{t'}\}, s^t) - \text{ExpUtil}_i^\pi(a^t, s^t) \end{aligned}$$

$$\begin{aligned} \text{ExpRegret}_i^\pi(a_i^t, a_i^{t'}, s^t) & \quad (3) \\ \triangleq \mathbb{E}_{A_{-i}^t} \left[\text{ExpDevGain}_i^\pi(a^t, s^t, a_i^{t'}) \middle| a_i^t, s^t \right]. \end{aligned}$$

Definition 3. A **correlated equilibrium (CE)** for a Markov game is a mixed joint strategy profile, π^{CE} , where

¹Markovian strategy profiles are a consequence of the MCECE formulation and commonly assumed in other solution concept formulations.

no expected gain is obtained for any agent by substituting an action, $a_i^{t'}$ that deviates from the strategy. This is guaranteed with the following set of constraints:

$$\forall_{t \in T, i \in N, s^t \in S, a_i^t \in S, a_i^{t'} \in S} \text{ExpRegret}_i^{\pi^{CE}}(a_i^t, a_i^{t'}, s^t) \leq 0. \quad (4)$$

CE (*Definition 3*) generalize **Nash equilibria** [22], which further require agents' actions in each state to be independent. Agents in a CE can coordinate their actions to obtain higher expected utilities. Conceptually, each agent is provided an action, a_i^t , and knows the conditional distribution of other agents' actions, $P(a_{-i}^t | a_i^t)$. To be in correlated equilibrium requires that no agent has an incentive to switch from action a_i^t to a deviation action, $a_i^{t'}$. Traffic lights are a canonical example of a **signaling device** designed to produce CE strategies. Given other agents' prescribed strategies (go on green), an agent will have incentive (equivalently, non-positive deviation regret) to obey its prescribed action (stop on red) rather than deviating (go on red). this coordination mechanism is not required as long as players have access to a public communications channel [6]. Past research has shown that many decentralized, adaptive strategies will converge to a CE [23, 10, 14, 11], and not necessarily to more restrictive equilibria, such as the Nash equilibrium.

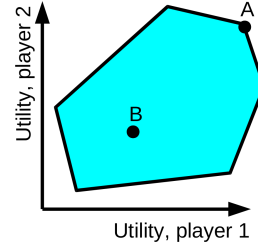


Figure 2: A CE polytope with a CE-Q equilibria (point A) maximizing average utility and a Max-EntCE (point B).

The deviation regret constraints (Equation 4) define an N -dimensional convex polytope of CE solutions in the space of agents' joint utility payoffs (Figure 2). Exactly representing this polytope is generally intractable for Markov games, because the number of corners of the polytope grows exponentially with the game's time horizon. Efficient approximation approaches have been employed [21, 5], but tractable applicability has been limited to small games (15 or fewer joint actions combinations per state) [5]. For the far more modest goal of finding an arbitrary CE in a range of compact games, algorithms that are polynomial in the number of agents have been developed [25, 18] and extended to sequential games [16].

Our objective is different; we desire neither the (approximate) entire convex polytope of CE strategy profiles nor an arbitrary CE strategy profile. Rather, we desire a single CE strategy profile with certain properties that are useful for predictive purposes. This can be approached using optimization techniques. For a single-shot (*i.e.*, **normal-form**) game, there are $O(N|A_i|^2)$ regret constraints that are linear in a total of $O(|A|)$ strategy variables, $\{\pi(a)\}_{a \in A}$, and CE solutions can be efficiently obtained by solving a linear

program or a convex program:

$$\begin{aligned} & \max_{\pi} f_0(\pi(A)) \text{ such that:} & (5) \\ & \forall_{i,a_i,a_i'} \sum_{a_{-i}} \pi(a) (\text{Utility}(\{a_{-i}, a_i'\}) - \text{Utility}(a)) \leq 0, \\ & \forall_a \pi(a) \geq 0, \text{ and } \sum_a \pi(a) = 1. \end{aligned}$$

depending on the objective function, f_0 .

A **correlated-Q equilibria** (CE-Q) [12] employs a linear or convex function of strategy probabilities for the selection metric objective of Equation 5 to obtain utility-unique strategy profiles². A number of objectives have been proposed:

- **Utilitarian** (u CE-Q) maximizes the sum of agents' utilities, $\sum_{i=1}^N \mathbb{E}[\text{Utility}_i(a)|\pi]$;
- **Dictatorial** (d CE-Q) maximizes a specific agent's utility, $\mathbb{E}[\text{Utility}_i(a)|\pi]$;
- **Republican** (r CE-Q) maximizes the highest agent's utility, $\max_i \mathbb{E}[\text{Utility}_i(a)|\pi]$; and
- **Egalitarian** (e CE-Q) maximizes lowest agent's utility, $\min_i \mathbb{E}[\text{Utility}_i(a)|\pi]$.

More generally, strategies that penalize one or more agents are also possible. For example, grim-trigger strategies have been recognized as viable sub-game strategies that disincentivize an agent's undesirable actions. Two punishment-based selection criteria that we consider in this work are:

- **Disciplinarian** (x CE-Q) minimizes a specific agent's utility, $\mathbb{E}[\text{Utility}_i(a)|\pi]$; and
- **Inegalitarian** (i CE-Q) maximizes utility differences between two (groups of) agents, $\mathbb{E}[\text{Utility}_i(a) - \text{Utility}_j(a)|\pi]$.

The strong assumptions about agents' preferences constrain CE-Q solutions to cover corners of the CE polytope (Figure 2).

2.2 Entropy, Prediction, and Gambling

Information theory provides powerful tools for constructing predictive probability distributions. One of its basic measures is **Shannon's information entropy**, $H(P) \triangleq -\sum_{x \in X} P(x) \log_2 P(x)$, which measures the uncertainty of distribution P . Information theory has many connections to problems in gambling. For example, the entropy of distribution P , and the exponential rate at which a gambler who knows P can expect his investment to grow are related by Theorem 4.

THEOREM 4 ([4]). *The doubling rate, which specifies the wealth growth rate, $O(2^{W(P,b)})$, for random outcomes distributed according to P with bets in proportion to b and payoff multipliers, o , such that $\forall_x o(x) \geq 1$ and $\sum_x o(x)^{-1} = 1$, is:*

$$W(P, b) = \sum_{x \in X} P(x) \log(b(x)o(x)).$$

It is maximized by $b(x)^ = P(x)$ for uniform odds and provides an optimal doubling rate, $W^*(P) = \log |X| - H(P)$.*

²Unique strategy profiles are not guaranteed by the CE-Q solution concept—multiple actions can provide the same agent utility vector. We ignore this ambiguity and employ a single CE-Q from the possible set.

More generally, a gambler (or predictor) may not know the distribution P , but instead knows some constraints that P satisfies. For example, linear equality constraints, $g(P) = 0$, and inequality constraints, $h(P) \leq 0$, are common.

Definition 5. *The principle of maximum entropy [17] prescribes the maximum entropy probability distribution subject to equality and inequality constraints:*

$$\operatorname{argmax}_P H(P) \text{ such that: } g(P) = 0 \text{ and } h(P) \leq 0.$$

The maximum entropy distribution (Definition 5) provides important predictive guarantees (Theorem 6).

THEOREM 6 ([13]). *The maximum entropy distribution minimizes the worst case predictive log-loss,*

$$\inf_{P(X)} \sup_{\tilde{P}(X)} - \sum_{x \in X} \tilde{P}(x) \log P(x),$$

subject to constraints $g(P) = 0$ and $h(P) \leq 0$.

Additionally, the gambling asset allocation that maximizes the worst-case growth-rate for this setting is:

$$b(X)^* = \operatorname{argmax}_{b(X)} \min_{P(X)} W(\mathbf{X}) \quad (6)$$

subject to equality and inequality constraints.

COROLLARY 7 (THEOREM 6 and THEOREM 4). *The optimal gambling asset allocation, $b(X)^*$, is proportional to the maximum entropy distribution when the payoff multipliers are uniform.*

2.3 Maximum Entropy Correlated Equilibria

The **maximum entropy correlated equilibria** (MaxEntCE) solution concept for normal-form games [24] selects the unique joint strategy profile that satisfies the principle of maximum entropy (Definition 5) subject to linear deviation regret inequality constraints (Equation 4). This approach provides the predictive and gambling guarantees of maximum entropy (THEOREM 6 and COROLLARY 7) to the one-shot, normal-form multi-agent game setting.

Table 1: The game of Chicken and four strategy profiles that are in correlated equilibrium.

	Stay	Sswerve		
Stay	0,0	4,1		
Sswerve	1,4	3, 3		
	<i>CE 1</i>	<i>CE 2</i>	<i>CE 3</i>	<i>CE 4</i>
0	1	0	0	$\frac{1}{3}$
0	0	1	0	$\frac{1}{3}$
			$\frac{1}{3}$	$\frac{1}{3}$
			$\frac{1}{4}$	$\frac{1}{4}$

Consider the game of Chicken (where each agent hopes the other will *Sswerve*) and the correlated equilibria that define its utility polytope in TABLE 1. We relate these strategy profiles to the more specific equilibria described in Section 2.1. *CE 1* and *CE 2* are both dictatorial, disciplinarian, and inegalitarian CE (for different agents) and republican CE (but ambiguous). *CE 3* is a utilitarian CE and an egalitarian CE. *CE 4* is the maximum entropy CE. Its predictive guarantee is apparent: all other CE have infinite log-loss for at least one other CE; the MaxEntCE is the only CE that assigns positive probability to the {Stay, Stay} action combination. We extend these predictive guarantees to the Markov setting in this work.

3. MAXIMUM CAUSAL ENTROPY CORRELATED EQUILIBRIA

Extension of the MaxEntCE solution concept [24] to the Markov game setting is not straight-forward. The first difficulty is that the deviation regret constraints of normal-form games (Equation 5), contain expectations over future actions (Equation 4) when extended to the Markov game setting. This creates non-linear constraints that are products of the unknown variables, making optimization difficult.

THEOREM 8. *A linear/convex program formulation of CE for Markov games is possible by considering as variables the entire sequence of joint agent actions for the sequence of revealed states, $\eta(A^{1:T}|S^{1:T})$, and employing appropriate inequality constraints (deviation regret guarantees) and equality constraints (forcing the strategy over sequences to factor into products of Markovian strategies) on marginal distributions using linear function of $\eta(A^{1:T}|S^{1:T})$ variables.*

Naïvely formulating the Markov game CE strategy profiles into a linear/convex program is possible (THEOREM 8), but the number of constraints and variables grow exponentially with the time horizon.

The second difficulty is that there are many entropy measures that could be applied as objective functions. For example, the **conditional entropy** and **joint entropy** are natural entropy measures to consider. However, neither appropriately extends the predictive and gambling guarantees of the maximum entropy approach to the sequential Markov game setting. They either assume the availability of future outcome information (violating the problem setting), or are not risk-neutral to the stochasticity of the Markov game's transition dynamics.

We instead advocate the less common **causally conditioned entropy** measure [19],

$$H(\mathbf{A}^T||\mathbf{S}^T) \triangleq \sum_t H(A^t|A^{1:t-1}, S^{1:t}). \quad (7)$$

For the possible sequences of states and actions through a Markov game, it corresponds to the uncertainty associated with only the actions in such sequences. It is based on the **causally conditioned probability distribution**, $P(\mathbf{A}^T||\mathbf{S}^T) \triangleq \prod_t P(A^t|A^{1:t-1}, S^{1:t})$, which conditions each set of correlated actions only on actions and states that have been revealed at that point in time and not on future states, as in the conditional probability distribution $P(\mathbf{A}|\mathbf{S}) = \prod_t P(A^t|A^{1:t-1}, S^{1:t}, S^{t+1:T})$.

Definition 9. A **maximum causal entropy correlated equilibrium** (MCECE) solution maximizes the causal entropy while being constrained to have no action deviation regrets³:

$$\begin{aligned} \pi^{MCECE} &\triangleq \underset{\pi}{\operatorname{argmax}} H(\mathbf{A}^T||\mathbf{S}^T) \\ &= \underset{\pi}{\operatorname{argmax}} \mathbb{E}_{A^{1:T}, S^{1:T}} \left[\sum_{t=1}^T -\log P(a^t|s^t) \right] \end{aligned} \quad (8)$$

³Markovian policies are a consequence of the MCECE formulation. See Lemma 13.

such that: $\forall_{t, i, a_i^t, a_i^{t'}, s^t} \operatorname{ExpRegret}_i^\pi(a_i^t, a_i^{t'}, s^t) \leq 0$,

$$\forall_{t, s^t, a^t} P(a^t|s^t) \geq 0, \quad \forall_{t, s^t} \sum_{a^t} P(a^t|s^t) = 1,$$

π factors as: $P(\mathbf{A}^T||\mathbf{S}^T)$ and given: $P(S^{t+1}|S^t, A^t)$.

We further constrain the strategy profile to have **sub-game equilibria**, meaning that even in states that are unreachable under the strategy profile and state dynamics, the strategy profile is constrained to satisfy Equation 8 in all sub-games starting from those states.

Maximizing the causal entropy (Equation 8) has been previously employed to match characteristics of demonstrated behavior in decision settings using equality constraints [26]. MCECE represents the first inequality-constrained application of the principle of maximum causal entropy

Based on the view of conditional entropy as a measure of predictability [4], the MCECE solution concept offers two important predictive guarantees:

THEOREM 10 (extension of [24]). *Given an MCECE strategy profile, no agent may decrease the predictability of her action sequence without creating deviation regret for herself.*

THEOREM 11 (extension of [13]). *The MCECE solution strategy profile, π^{MCECE} minimizes the worst-case log prediction loss for the sequences of joint actions, i.e.,*

$$\inf_{P(\mathbf{A}^T||\mathbf{S}^T)} \sup_{\tilde{P}(\mathbf{A}^T||\mathbf{S}^T)} - \sum_{\mathbf{a} \in \mathbf{A}, \mathbf{s} \in \mathbf{S}} \tilde{P}(\mathbf{a}, \mathbf{s}) \log P(\mathbf{a}^T||\mathbf{s}^T), \quad (9)$$

of all the CE satisfying deviation regret constraints, where $\tilde{P}(\mathbf{A}^T||\mathbf{S}^T)$ is the (worst possible for prediction) empirical CE strategy and the joint, $\tilde{P}(\mathbf{A}, \mathbf{S})$, is the distribution of states and actions under that strategy profile and the known state transition dynamics.

The second result (THEOREM 11) is particularly relevant to our machine learning perspective, because it justifies the MCECE strategy profile as a robust predictive model of agents' actions when they jointly behave rationally.

4. CORRELATED EQUILIBRIA FINDING

We turn our attention from the theoretical properties of the MCECE solution concept to developing an algorithm that obtains the MCECE for a fixed-horizon Markov game more efficiently than the naïve convex optimization that follows the formulation of Theorem 8. Despite the non-compact formulation of the naïve MCECE convex program, the strategy profile can be expressed compactly.

LEMMA 12. *The MCECE strategy profile for a Markov game is also Markovian.*

Proof (sketch). Intuitively, independence maximizes entropy. Since the utility structure and game dynamics are history-independent, nothing prevents the MCECE from also being history-independent. \square

THEOREM 13. *The MCECE strategy profile, $\pi_\lambda^{MCECE}(a^t|s^t)$, has the following recursive form (with $\lambda \geq 0$):*

$$\begin{aligned} \pi_\lambda(a^t|s^t) &\propto e^{-\left(\sum_{i, a_i^{t'}, \lambda_{i, s^t, a_i^t, a_i^{t'}} \operatorname{ExpDevGain}_i^\pi(a^t, s^t, a_i^{t'})\right)} \\ &\quad + \operatorname{ExpEnt}(a^t, s^t), \end{aligned} \quad (10)$$

where $\text{ExpEnt}(a^t, s^t) \triangleq \mathbb{E}_{P(A^{t+1}, S^{t+1})} [\text{ExpEnt}(a^{t+1}, s^{t+1}) + H(a^{t+1}|s^{t+1})|a^t, s^t]$.

We obtain optimal Lagrange variables, $\{\lambda_{i,s,a_i,a'_i}^*\} \geq 0$, by optimizing the Lagrange dual,

$$L_D(\lambda) = \mathbb{E}_{A^{1:T}, S^{1:T}} \left[-\log P_\lambda(a^{1:T} || s^{1:T}) \right] \quad (11)$$

$$- \sum_{t,i,s^t,a_i^t,a_i'^t} \lambda_{t,i,s^t,a_i^t,a_i'^t} \text{ExpRegret}_i^{\pi_\lambda}(a_i^t, a_i'^t, s^t),$$

using gradient-based optimization and the dual's gradient,

$$\nabla_\lambda L_D(\lambda) = \left\{ \sum_{a_{-i}^t} P(a^t | s^t) \left(\text{ExpUtil}_i^{\pi_\lambda}(\{a_{-i}^t, a_i'^t\}, s^t) - \text{ExpUtil}_i^{\pi_\lambda}(a^t, s^t) \right) \right\}, \quad (12)$$

as shown in Algorithm 1.

Algorithm 1 Find MCECE equilibria for finite horizon

- 1: $\lambda^{(1)} = \{\lambda_{t,i,s^t,a_i^t,a_i'^t}^{(1)}\} \leftarrow$ (arbitrary) positive initial values.
 - 2: $x \leftarrow 1$
 - 3: **while** not converged **do**
 - 4: Compute $\pi_\lambda^{(x)} = \{\pi_\lambda(a^t | s^t)\}$ from $\lambda^{(x)}$ using a subroutine.
 - 5: Compute $L_D(\lambda^{(x)})$ and $\nabla_\lambda L_D(\lambda^{(x)})$ directly via Equation 11 and Equation 12 using $\pi_\lambda^{(x)}$
 - 6: Update $\lambda^{(x+1)}$ from $\{\lambda, L_D, \nabla_\lambda L_D\}^{(1:x)}$ using gradient-based optimization update rules
 - 7: $x \leftarrow x + 1$
 - 8: **end while**
-

REMARK 14. *For time-varying policies, future strategy probabilities (and dual parameters) are independent of earlier strategy and dual parameters given the state. As a result, the “parallel” updates for dual parameters across time (Algorithm 1) can be sequentially ordered for improved efficiency.*

Following Remark 14, Algorithm 1 can be re-expressed as a sequential dynamic programming algorithm (Algorithm 2) resembling value iteration [2] that iteratively computes both future expected utilities and expected entropies. It also suggests the parallel updating of the dual parameters (or the primal policy) as a general approach for overcoming the limitations of value iteration for finding stationary CE strategy profiles in general-sum Markov games. However, full discussion is beyond this paper's scope.

Using interior-point methods, an ϵ -optimal MCECE strategy profile is obtained in $O(|S|^{\frac{T}{2}} |A|^{\frac{T}{2}} \log \frac{1}{\epsilon})$ time using the naïve formulation of Theorem 8. Using Algorithm 2, this is reduced to $O(|S|T|A|^{\frac{1}{2}} \log \frac{|S|T}{2\epsilon})$ time.

We employ existing sub-gradient optimization methods [3] for the convex objective and linear inequality constraints to obtain the strategy profiles for the interior optimization (Line 4) of Algorithm 2. This provides looser runtime bounds than interior-point optimization methods, but is simpler to implement and still practical for the purposes of this paper.

Algorithm 2 Value iteration approach for obtaining MCECE

- 1: $\forall_{i,a,s} \text{ExpUtil}_i(a, s) \leftarrow \text{Utility}_i(a, s)$
 - 2: $\forall_{a,s} \text{ExpEnt}(a, s) \leftarrow 0$
 - 3: **for** $t = T$ to 1 **do**
 - 4: For each state, s^t , obtain $\{\pi_\lambda(a^t | s^t)\}$ using ExpUtil and ExpEnt values in the following optimization:

$$\underset{\pi(a^t | s^t)}{\text{argmax}} H(a^t | s^t) + \mathbb{E} [\text{ExpEnt}(a, s) | s^t, \pi(a^t | s^t)]$$
 such that: $\sum_{a_{-i} \in A_{-i}} P(a^t | s^t) \left(\text{ExpRegret}(\{a_{-i}^t, a_i^t\}, s^t) - \text{ExpRegret}(a^t, s^t) \right) \leq 0$
 $\forall_{a^t} P(a^t | s^t) \geq 0$ and $\sum_{a^t \in A^t} P(a^t | s^t) = 1.$
 - 5: $\forall_{i \in N, a \in A, s \in S} \text{ExpUtil}'_i(a, s) \leftarrow \gamma \sum_{a^t \in A, s^t \in S} \pi(a^t | s) P(s^t | s, a) \text{ExpUtil}_i(a^t, s^t)$
 - 6: $\forall_{s \in S, a \in A} \text{ExpEnt}'(a, s) \leftarrow \gamma \sum_{a^t, s^t} \pi(a^t | s^t) P(s^t | s, a) (\text{ExpEnt}(a', s') + H(a' | s'))$
 - 7: $\forall_{i \in N, a \in A, s \in S} \text{ExpUtil}_i(a, s) \leftarrow \text{ExpUtil}'_i(a, s) + \text{Utility}_i(a, s)$
 - 8: $\forall_{a \in A, s \in S} \text{ExpEnt}(a, s) \leftarrow \text{ExpEnt}'(a, s)$
 - 9: **end for**
-

5. EXPERIMENTAL EVALUATION

In this section, we demonstrate that the theoretical robust predictive guarantees of the MCECE are realized in practice. Following Zinkevich et al. [27], we generate random Markov games for evaluation. We compute different strategy profiles for each generated game using existing CE solution concepts and evaluate how well they predict one another.

5.1 Setup

We generate random stochastic Markov games according to the following procedure. For each of $|S|$ states in the Markov game, each agent has $|A_i|$ actions from which to choose, and there are $|A|$ joint actions total. The state transition dynamics, $P(S_{t+1} | S_t, A_t)$, depend on the combination of agents' actions (and state) and are drawn uniformly from the simplex of probabilities. The utility obtained by each agent in each state, $\text{Utility}_i(s)$, is drawn uniformly from $\{0, 0.1, 0.2, \dots, 0.9\}$. A discount factor of $\gamma = .75$ is incorporated in each game. It's important to note that we did not optimize these random game parameters to obtain desired results; we expect the results for the games we evaluate to extend to a wide range of games—random or otherwise.

We generate time-varying strategy profiles for MCECE using Algorithm 2 and for the CE-Q variants using projected sub-gradient optimization. The CE-Q strategies we evaluate are a subset of those described in Section 2.1. i C-EQ maximizes the positive margin of agent 1's utility over agent 2's utility. We repeat this process for 100 random games for each choice of game parameters and investigate the properties of the resulting CE strategy profiles.

As shown in Figure 3, the uncertainty of action sequences increases linearly with the size of the action set, as one might expect. Previous experiments have primarily considered two-player Markov games [12, 27]. The larger number of players we consider in this paper greatly increases the game complexity since the game description grows exponen-

Table 2: Predictive bake-off evaluation of the first action in a ten timestep horizon using 100 random Markov games. Log-loss and non-support measures (equivalent to total gambling loss) are evaluated.

Seven equilibria strategy profiles evaluated on random Markov games with **three** agents, two states, and two actions/agent.

	MCECE	u CE-Q	d_1 CE-Q	d_2 CE-Q	x_1 CE-Q	x_2 CE-Q	i CE-Q	Average
MCECE	—	1.951 0.0%	1.951 0.0%	1.967 0.0%	2.010 0.0%	1.992 0.0%	1.974 0.0%	1.974 0.0%
u CE-Q	3.377 22.3%	—	2.039 4.5%	1.888 5.7%	2.647 21.2%	3.072 20.8%	2.444 13.8%	2.578 14.7%
d_1 CE-Q	3.442 18.9%	1.866 3.5%	—	2.511 5.6%	3.328 18.8%	2.321 17.6%	1.798 9.8%	2.544 12.4%
d_2 CE-Q	3.462 17.0%	1.872 1.7%	2.536 3.3%	—	2.576 15.6%	3.489 17.2%	3.060 12.1%	2.833 11.2%
x_1 CE-Q	2.897 0.2%	2.472 0.0%	2.798 0.0%	2.450 0.0%	—	2.375 0.0%	2.764 0.0%	2.626 0.0%
x_2 CE-Q	2.877 0.5%	2.605 0.0%	2.251 0.0%	2.905 0.0%	2.373 0.2%	—	2.116 0.0%	2.521 0.1%
i CE-Q	3.378 5.3%	2.279 1.9%	1.902 0.0%	2.989 2.5%	3.116 5.1%	2.276 4.2%	—	2.657 3.2%

Seven equilibria strategy profiles evaluated on random Markov games with **four** agents, two states, and two actions/agent.

	MCECE	u CE-Q	d_1 CE-Q	d_2 CE-Q	x_1 CE-Q	x_2 CE-Q	i CE-Q	Average
MCECE	—	3.451 0.0%	3.468 0.0%	3.476 0.0%	3.518 0.0%	3.509 0.0%	3.475 0.0%	3.483 0.0%
u CE-Q	7.308 25.8%	—	3.955 9.4%	3.652 8.5%	6.495 21.8%	6.814 22.0%	5.591 17.9%	5.636 17.6%
d_1 CE-Q	6.914 22.5%	3.831 5.8%	—	4.746 10.2%	7.566 21.2%	5.536 17.6%	3.895 11.2%	5.415 14.8%
d_2 CE-Q	7.109 23.1%	3.408 6.4%	4.767 10.7%	—	5.643 18.3%	7.651 21.5%	6.976 19.7%	5.926 16.6%
x_1 CE-Q	4.603 0.0%	4.300 0.0%	5.000 0.0%	3.849 0.0%	—	3.918 0.0%	4.372 0.0%	4.340 0.0%
x_2 CE-Q	4.633 0.1%	4.401 0.0%	3.917 0.0%	4.986 0.0%	3.944 0.0%	—	3.171 0.0%	4.175 0.0%
i CE-Q	6.380 11.5%	5.070 5.0%	3.884 2.8%	6.501 8.0%	5.991 8.7%	4.311 5.5%	—	5.356 6.9%

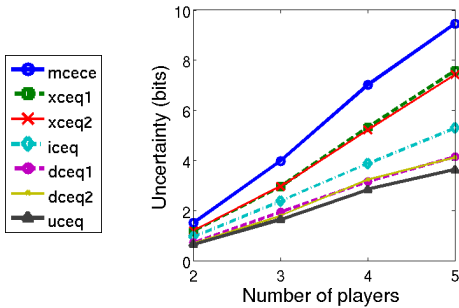


Figure 3: The average causal entropy measure of the 10 time step action sequences that are generated by different correlated equilibria solution concepts for random 2-state, 2-action Markov games.

tially with the number of players. In our experiments, the MCECE strategy profile is the most uncertain (by design) and many of the previously investigated CE-Q solutions are the most deterministic.

5.2 Evaluation Metrics

The predictive guarantees of the MCECE strategy profiles relate to the log-loss of predicting actions distributed according to P_{obs} with distribution P_{pred} :

$$-\sum_{a \in A} P_{\text{obs}}(a) \log P_{\text{pred}}(a).$$

In the context of this work, P_{obs} and P_{pred} are each proba-

bilities in CE strategy profiles for Markov games. Unfortunately, many of the strategy profiles provide no support for some action combinations that are possible in other strategy profiles. In other words, they predict that some action combinations occur with 0% probability when they do in fact occur with positive probability. This corresponds to an infinite log-loss.

Instead of using the typical log-loss measures, which is often infinite except for the MCECE strategy profile, we instead employ two measures. The first is the log-loss on the action combinations that do have support⁴. The second is the percentage of action combinations that have no support. The latter can be interpreted as the degree of infiniteness that the log loss would have. Equivalently, under the gambling perspective, it can be interpreted as the percentage of instances all of a gambler’s money would be lost.

5.3 Action Prediction Comparison

The results of this comparison across strategy profiles are shown in TABLE 2 (for three and four agents). We note that in some cases one C-EQ strategy profile may better predict another than the MCECE strategy profile when their objectives are closely aligned. For example, the x_2 CE-Q predicts i CE-Q fairly accurately since both are punishing agent 2 to some degree. However, overall the MCECE solution profile provides a much more robust prediction of other strategy profiles (and full support) on average (right column).

We employ the relationships between log-loss and dou-

⁴To address ϵ approximation error, we employ a minimum P_{obs} threshold of 0.1% for assessing non-support and a maximum penalty threshold of 16.6 bits ($-\log_2 0.00001$) for small support—both to the benefit of CE-Q strategy profiles.

Table 3: Doubling rates of CE as gambling allocations in the three and four player game settings.

	Three players	Four players
MCECE	1.026	0.517
μ CE-Q	0.422	-1.636
d_1 CE-Q	0.456	-1.415
d_2 CE-Q	0.167	-1.926
x_1 CE-Q	0.374	-0.346
x_2 CE-Q	0.479	-0.175
i CE-Q	0.343	-1.356

bling rate from Section 2.2 to illustrate the benefits of different CE for gambling in Table 3. For the three player experimental setting, all CE strategies are expected to have positive investment growth rate (under uniform, fair odds). However, many also have a probability of losing all money (Table 2). Betting according to the MCECE distribution provides the largest growth rate—with the expectation of doubling an investment after each bet. For the four player setting, the MCECE distribution is the only one with positive expected investment growth. Thus, the theoretical properties for prediction under log-loss and gambling under uniform odds provided by the MCECE are realized in practice.

6. CONCLUSIONS

This paper was motivated by a fundamental question: given that agents act rationally (*i.e.*, according to an unknown correlated equilibrium) within a known Markov game, and no other information is available, what predictions of agents’ action sequences should be employed? We employed an extension of information theory and the principle of maximum entropy to develop a predictive solution concept that addresses this question. We demonstrated the robustness of its predictions across a wide range of existing value-based CE solution concepts. In many settings where decisions are made based on the action sequences of self-interested, communicating autonomous agents, this assumption is reasonable. We have shown in theory and in practice the predictive guarantees of the approach and connected its guarantees to the gambling setting. Generalizing this approach to extensive form games is of interest, however it introduces non-convexity. We could replace the joint entropy measure of past coordinate descent approaches with the causal entropy measure [7].

Our view in this paper has been agnostic, apart from assuming joint rationality. Often, past agent behavior may be known. Important future work extends this approach to when such additional information *is* available. This can be accomplished with the addition of behavior-matching equality constraints to the MCECE solution concept optimization. Additionally, since actual behavior may only be approximately jointly rational, relaxing the inequality constraints using dual regularization [8] is another important direction. Extending the maximum entropy approach to behavior that is guided by unknown utility functions remains as an important future problem.

Acknowledgments

The authors gratefully acknowledge the Richard K. Mellon Foundation, the Quality of Life Technology Center, and the

Office of Naval Research Reasoning in Reduced Information Spaces project MURI for support of this research. We thank Geoff Gordon and Miro Dudík for useful discussions.

7. REFERENCES

- [1] R. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, 1(1):67–96, 1974.
- [2] R. Bellman. A Markovian decision process. *Journal of Mathematics and Mechanics*, 6:679–684, 1957.
- [3] S. Boyd, L. Xiao, and A. Mutapcic. Subgradient methods. *Lecture notes of EE392o, Stanford University, Autumn Quarter*, 2003.
- [4] T. Cover and J. Thomas. *Elements of information theory*. Wiley, 2006.
- [5] L. M. Dermed and C. L. Isbell. Solving Stochastic Games. In *Proc. NIPS*, pages 1186–1194, 2009.
- [6] Y. Dodis, S. Halevi, and T. Rabin. A cryptographic solution to a game theoretic problem. In *Advances in Cryptology*, pages 112–130. Springer, 2000.
- [7] M. Dudík and G. Gordon. A sampling-based approach to computing equilibria in succinct extensive-form games. In *Proc. UAI*, pages 151–160, 2009.
- [8] M. Dudík and R. E. Schapire. Maximum entropy distribution estimation with generalized regularization. In *Proc. COLT*, pages 123–138, 2006.
- [9] S. Ficci and A. Pfeffer. Modeling how humans reason about others with partial information. In *Proc. AAMAS*, pages 315–322, 2008.
- [10] D. Foster and R. Vohra. Calibrated Learning and Correlated Equilibrium. *Games and Economic Behavior*, 21(1-2):40–55, 1997.
- [11] G. Gordon, A. Greenwald, and C. Marks. No-regret learning in convex games. In *Proc. ICML*, pages 360–367. ACM, 2008.
- [12] A. Greenwald and Hall. Correlated Q-learning. In *Proc. ICML*, pages 242–249, 2003.
- [13] P. D. Grünwald and A. P. Dawid. Game theory, maximum entropy, minimum discrepancy, and robust Bayesian decision theory. *Annals of Statistics*, 32:1367–1433, 2003.
- [14] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.
- [15] J. Hu and M. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. ICML*, pages 242–250, 1998.
- [16] W. Huang and B. von Stengel. Computing an extensive-form correlated equilibrium in polynomial time. *Internet and Network Economics*, pages 506–513, 2008.
- [17] E. T. Jaynes. Information theory and statistical mechanics. *Physical Review*, 106:620–630, 1957.
- [18] S. Kakade, M. Kearns, J. Langford, and L. Ortiz. Correlated equilibria in graphical games. In *Proc. Electronic Commerce*, pages 42–47. ACM, 2003.
- [19] G. Kramer. *Directed Information for Channels with Feedback*. PhD thesis, Swiss Federal Institute of Technology (ETH) Zurich, 1998.
- [20] M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. ICML*, pages 157–163, 1994.
- [21] C. Murray and G. Gordon. Multi-robot negotiation: approximating the set of subgame perfect equilibria in general-sum stochastic games. In *Proc. NIPS*, pages 1001–1008, 2007.
- [22] J. Nash. Non-cooperative games. *Annals of mathematics*, 54(2):286–295, 1951.
- [23] Y. Nyarko. Bayesian learning leads to correlated equilibria in normal form games. *Economic Theory*, 4(6):821–841, 1994.
- [24] L. E. Ortiz, R. E. Schapire, and S. M. Kakade. Maximum entropy correlated equilibria. In *Proc. AISTATS*, pages 347–354, 2007.

- [25] C. Papadimitriou and T. Roughgarden. Computing equilibria in multi-player games. In *Proc. SODA*, pages 82–91, 2005.
- [26] B. D. Ziebart, J. A. Bagnell, and A. K. Dey. Modeling interaction via the principle of maximum causal entropy. In *Proc. ICML*, pages 1255–1262, 2010.
- [27] M. Zinkevich, A. Greenwald, and M. Littman. Cyclic equilibria in Markov games. In *Proc. NIPS*, volume 18, pages 1641–1648, 2006.

APPENDIX

A. PROOFS OF THEOREMS

THEOREM 8 (Proof). Consider optimizing over conditional strategy sequence variables, $\eta(a^{1:T}|s^{1:T})$, that represent the probability of an entire sequence of actions given the entire sequence of states. Action-state strategy probabilities, $\pi(a^t|s^{t:T})$, can be obtained by marginalizing over a linear function of conditional sequence variables:

$$\begin{aligned} \pi(a^t|s^{t:T}) &= \sum_{s^{1:t-1}} \sum_{a^{1:t-1}} \sum_{a^{t+1:T}} P(a^{1:T}, s^{1:t-1}|s^{t:T}) \quad (13) \\ &= \sum_{s^{1:t-1}} \sum_{a^{1:t-1}} \sum_{a^{t+1:T}} \eta(a^{1:T}|s^{1:T}) \prod_{\tau=1}^{t-1} P(s^\tau|a^{\tau-1}, s^{\tau-1}). \end{aligned}$$

Crucially, to match the Markov game setting, the conditional distribution of actions at time step t should be equivalent regardless of future state variables, $s^{t+1:T}$, since those variables are not yet known in the Markov game:

$$\begin{aligned} \forall_{t,a^t,s^t,s^{t+1:T},\tilde{s}^{t+1:T}} \\ \pi(a^t|s^t, s^{t+1:T}) = \pi(a^t|s^t, \tilde{s}^{t+1:T}). \quad (14) \end{aligned}$$

The constraints (Equation 14) are linear of conditional strategy sequence variables via the steps of Equation 13.

The expected regret can similarly be expressed as a linear function of conditional strategy sequence variables:

$$\begin{aligned} \text{ExpRegret}_i^\pi(a_i^t, a_i^{t'}, s^t) \\ &= \sum_{a_{-i}^t, a^{t+1:T}, s^{1:t-1}, s^{t+1:T}} \eta(a^{1:T}|s^{1:T}) \frac{\prod_{\tau=1}^T P(s^{\tau+1}|s^\tau, a^\tau)}{P(s^{t+1}|s^t, a^t)} \times \\ &\left(P(s^{t+1}|s^t, a^{t'}) \left(\sum_{\tau>t} \text{Util}(s^\tau, a^\tau) + \text{Util}(s^t, a^{t'}) \right) \right. \\ &\left. - P(s^{t+1}|s^t, a^t) \left(\sum_{\tau>t} \text{Util}(s^\tau, a^\tau) + \text{Util}(s^t, a^t) \right) \right) \end{aligned}$$

All constraints are linear in conditional variables, so when $-f_0$ is a linear or convex function, the MCECE optimization (Equation 15) is a linear program or convex program.

$$\text{argmax}_{\{\eta(a^{1:T}|s^{1:T})\}} f_0(\{\eta(a^{1:T}|s^{1:T})\}) \quad (15)$$

$$\text{such that: } \forall_{t,i,s^t,a_i^t,a_i^{t'}} \text{ExpRegret}_i^\pi(a_i^t, a_i^{t'}, s^t) \leq 0$$

$$\forall_{t,s^t,a^t} \pi(a^t|s^{t:T}) \geq 0, \quad \forall_{t,s^t} \sum_{a^t} \pi(a^t|s^{t:T}) = 1$$

$$\forall_{t,a^t,s^t,s^{t+1:T},\tilde{s}^{t+1:T}} \pi(a^t|s^t, s^{t+1:T}) = \pi(a^t|s^t, \tilde{s}^{t+1:T}).$$

This formulation has $O(T|S|^T|A|)$ non-redundant constraints and a total of $O(|S|^T|A|^T)$ variables. \square

THEOREM 10 (Proof). Ignoring all the deviation regret constraints in our notation, consider the decomposition of the

causally conditioned entropy using the chain rule:

$$\begin{aligned} &\text{argmax}_{\{\pi(a^t|s^t)\}} H(A^T||S^T) \\ &= \text{argmax}_{\{\pi(a^t|s^t)\}} \left(H(A_i^T||S^T) + H(A_{-i}^T||S^T) \right) \\ &= \left\{ \pi^{\text{MCECE}}(a_{-i}^t|s^t) \right\} \cup \text{argmax}_{\{\pi(a_i^t|s^t)\}} H(A_i^T||S^T, A_{-i}^T). \end{aligned}$$

As shown, this is equivalent to a causally conditioned entropy maximization of agent i 's strategy profile (with the suppressed deviation regret constraints) given the combined MCECE strategy profile of the other agents. By definition this is the least predictable strategy profile that agent i can employ (subject to any deviation regret constraints). \square

THEOREM 11 (Proof sketch). As a special case of [13], the causal entropy can be expressed as:

$$\begin{aligned} H(\tilde{P}(\mathbf{A}^T||\mathbf{S}^T)) &= \inf_{P(\mathbf{A}^T||\mathbf{S}^T)} \mathbb{E}_{\tilde{P}(\mathbf{A},\mathbf{S})}[-\log P(\mathbf{A}^T||\mathbf{S}^T)]. \\ \text{Choosing } \tilde{P}(\mathbf{Y}^T||\mathbf{X}^T) &\text{ that maximizes this is then:} \\ \sup_{\tilde{P}(\mathbf{A}^T||\mathbf{S}^T)} \inf_{P(\mathbf{A}^T||\mathbf{S}^T)} \mathbb{E}_{\tilde{P}(\mathbf{A},\mathbf{S})}[-\log P(\mathbf{A}^T||\mathbf{S}^T)], &\text{ which is invariant to swapping the sup and inf operation order. } \square \end{aligned}$$

THEOREM 13 (Proof). We find the form of the probability distribution by finding the optimal point of the Lagrangian.

$$\text{argmax}_{\pi} H(\mathbf{A}^T||\mathbf{S}^T) \text{ such that:} \quad (16)$$

$$\begin{aligned} \forall_{t,i,a_i^t,a_i^{t'},s^{1:t},a^{1:t-1}} \text{ExpRegret}_i^\pi(a_i^t, a_i^{t'}, s^{1:t}, a^{1:t-1}) \leq 0 \\ \text{and probabilistic/causal constraints on } \pi. \end{aligned}$$

The Lagrangian for the optimization of Equation 16 when using entire history-dependent probability distributions and parameters is:

$$\begin{aligned} \Lambda(\pi, \lambda) &= H(a^{1:T}||s^{1:T}) \\ &- \sum_{t,i,a_i^t,a_i^{t'},s^{1:t},a^{1:t-1}} \lambda_{t,i,a_i^t,a_i^{t'},s^{1:t},a^{1:t-1}} \text{ExpRegret}_i^\pi(a_i^t, a_i^{t'}, s^{1:t}, a^{1:t-1}) \quad (17) \end{aligned}$$

Taking the partial derivative with respect to a history-dependent action probability for a particular state, we have:

$$\begin{aligned} \frac{\partial \Lambda(\pi, \lambda)}{\partial P(a^t|s^{1:t}, a^{1:t-1})} &= -\log P(a^t|s^{1:t}, a^{1:t-1}) \\ &- \sum_{s^{t+1:T}, a^{t+1:T}} P(s^{t+1:T}, a^{t+1:T}) \log \prod_{\tau=t}^T P(a^\tau|s^{1:\tau}, a^{1:\tau-1}) \\ &- \sum_{i,a_i^{t'}} \lambda_{t,i,a_i^t,a_i^{t'},s^{1:t},a^{1:t-1}} \text{ExpDevGain}_i^\pi(a_i^t, a_i^{t'}, s^{1:t}, a^{1:t-1}). \quad (18) \end{aligned}$$

Equating Equation 18 to zero provides the form of the history dependent distribution:

$$P(a^t|s^{1:t}, a^{1:t-1}) \propto \exp \left\{ \quad (19) \right.$$

$$\begin{aligned} &\sum_{s^{t+1:T}, a^{t+1:T}} P(s^{t+1:T}, a^{t+1:T}) \log \prod_{\tau=t}^T P(a^\tau|s^{1:\tau}, a^{1:\tau-1}) \\ &\left. - \sum_{i,a_i^{t'}} \lambda_{t,i,a_i^t,a_i^{t'},s^{1:t},a^{1:t-1}} \text{ExpDevGain}_i^\pi(a_i^t, a_i^{t'}, s^{1:t}, a^{1:t-1}) \right\}. \end{aligned}$$

Convex duality in this optimization relies on a feasible solution on the relative interior of the constraint set. This can be accomplished by adding an infinitesimally small amount of slack, ϵ , to the constraint set [24]. Following the argument that the MCECE is Markovian (Lemma 12), Equation 19 reduces to the Markovian form of the theorem. \square

Multiagent Learning

Learning Action Models for Multi-Agent Planning

Hankz Hankui Zhuo
Dept of Computer Science,
Sun Yat-sen University,
Guangzhou, China.
zhuohank@mail.sysu.edu.cn

Hector Muñoz-Avila
Dept of Computer Science &
Engineering,
Lehigh University,
Bethlehem, PA, USA
munoz@cse.lehigh.edu

Qiang Yang
Dept of Computer Science &
Engineering,
Hong Kong University of
Science and Technology,
Kowloon, Hong Kong.
qyang@cse.ust.hk

ABSTRACT

In multi-agent planning environments, action models for each agent must be given as input. However, creating such action models by hand is difficult and time-consuming, because it requires formally representing the complex relationships among different objects in the environment. The problem is compounded in multi-agent environments where agents can take more types of actions. In this paper, we present an algorithm to learn action models for multi-agent planning systems from a set of input plan traces. Our learning algorithm *Lammas* automatically generates three kinds of constraints: (1) constraints on the interactions between agents, (2) constraints on the correctness of the action models for each individual agent, and (3) constraints on actions themselves. *Lammas* attempts to satisfy these constraints simultaneously using a weighted maximum satisfiability model known as MAX-SAT, and converts the solution into action models. We believe this to be one of the first learning algorithms to learn action models in the context of multi-agent planning environments. We empirically demonstrate that *Lammas* performs effectively and efficiently in several planning domains.

Categories and Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning - Knowledge acquisition;
I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence - Intelligent agents, Multiagent systems.

General Terms

Algorithms, Performance, Experimentation.

Keywords

Multi-Agent Planning, Multi-Agent Learning, Single-Agent Learning.

1. INTRODUCTION

Multi-agent environments are complex domains in which agents aim at pursuing their goals while interacting with each other. For multi-agent planning, each agent requires an action model as input that takes into account the possible prerequisites and outcomes, as well as interactions with other agents. For example, an agent ϕ_i needs to consider many complex situations where *cooperative* agents provide conditions such that ϕ_i 's actions can be executed.

Cite as: Learning Action Models for Multi-Agent Planning, Hankz Hankui Zhuo, Hector Muñoz-Avila and Qiang Yang, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 217-224.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Furthermore, the action model should allow *cooperative* agents to delete conditions as side-effects when providing useful preconditions, and be able to represent other *non-cooperative* agents that might interfere with ϕ_i 's action. Creating action models for these agents by hand is difficult and time-consuming due to the complex interactions among agents.

Our objective is to explore learning algorithms that can automatically learn action models in multi-agent environments that can then be fed to multi-agent planning systems, such as the *planning first* system [11]. In the past, there have been several works on learning action models for single agents, such as ARMS [16] and SLAF [1]. However, these learning algorithms did not take into account multi-agent situations. One possibility in tackling this multi-agent learning problem is to assume that there is an *oracle agent* that knows and executes all the actions of the agents. In this situation, we can learn the action models for the oracle agent by using a *single-agent* learning algorithm, such as ARMS ([16]). This approach, however, neglects to consider the interactions between the agents and, as a result, may increase the errors of the learned models due to the potentially large number of interactions among the agents.

In this paper, we present a novel multi-agent action-model learning system known as *Lammas*. *Lammas* stands for (**L**earning **A**ction **M**odels for **M**ulti-**A**gent **S**ystems). In order for *Lammas* to explicitly capture the interactions between agents, *Lammas* generates and exploits an agent-interaction graph in which it captures the interactions between pairs of agents. Such interactions may happen when one agent's action provides some positive, or negative, effects on the actions of other agents. For instance, consider a domain where there are two agents *truck* and *hoist*, where the action "drive" of agent *truck* provides an effect "(at truck loc)" for the action "load" of agent *hoist*, such that agent *hoist* can load a package to the "truck" at location "loc". In this example, the interactions can be somewhat complex because the interactions are *problem-specific*; i.e., building such interactions requires to explore all possible potential interactions among agents, and this is difficult to do for a human designer when there are many agents involved. To solve this problem, *Lammas* builds the relations statistically from a training data set that consists of *plan traces* from observed multi-agent plan executions in the past. These built relations help discover agent interactions that can then be transformed to weighted constraints and used in learning (Section 4).

For modeling the agents' actions, in this work we adopt a deterministic state-transition model expressed via the STRIPS planning representation language [4], slightly extended to associate actions with agents (i.e., each action is annotated with the agent that performs it). This extended language is called MA-STRIPS [2]. In *Lammas*, we first build three types of constraints from *plan traces* collected from multi-agent environments. The first type of con-

straints encodes the interactions among agents. The second type of constraint encodes the correctness requirements of plans for each agent. The third type encodes the constraints of actions required by STRIPS for each agent. We then satisfy these constraints simultaneously using a weighted maximum satisfiability (MAX-SAT) solver, and transform the solution into action models of each agent.

We organize the paper as follows. We first review related works in single and multi-agent planning area. Then, we present the formalities for our work, and give a detailed description of our LAMMAS algorithm. Finally, we empirically evaluate LAMMAS in several planning domains and conclude our work with a discussion on future works.

2. RELATED WORK

In this section we review previous works on multi-agent planning, learning action models, and multi-agent learning.

2.1 Multi-Agent Planning

Our work is related to multi-agent planning. In [6], Georgeff presented a theory of action for reasoning about events in multi-agent or dynamically-changing environments. Wilkins and Myers presented a multi-agent planning architecture for integrating diverse technologies into a system capable of solving complex planning problems [14]. Brafman and Domshlak (2008) established an exponential upper bound on the complexity of multi-agent planning problems depending on two parameters quantifying the level of agents' coupling [2]. They quantified the notion of agents' coupling and present a multi-agent planning algorithm that scales polynomially with the size of the problem for fixed coupling levels. Based on this work, Nissim *et al.* (2010) presented a distributed multi-agent planning algorithm [11]. They used distributed constraint satisfaction (CSP) to coordinate between agents and local planning to ensure consistency of the coordination points. To solve the distributed CSP efficiently, they modify some existing methods to take advantage of the structure of the underlying planning problem.

2.2 Action Model Learning

Another related work is action model learning for planning. Gil (1994) described a system called EXPO, which learns by bootstrapping an incomplete STRIPS-like domain description augmented with past planning experiences [7]. Wang (1995) proposed an approach to automatically learn planning operators by observing expert solution traces and refining the operators through practice in a learning-by-doing paradigm [13]. Holmes and Isbell, Jr. (2004) modeled synthetic items based on experience to construct action models [9]. Walsh and Littman (2008) presented an efficient algorithm for learning action schemas for describing Web services [12]. ARMS automatically learns action models from a set of observed plan traces [16] using MAX-SAT. Amir (2005) presented a tractable, exact solution SLAF for the problem of identifying actions' effects in partially observable STRIPS domains [1]. Cresswell *et al.* (2009) developed a system called LOCM designed to carry out automated induction of action models from sets of example plans. Compared with previous systems, LOCM learns action models with action sequences as input, and is shown to work well under the assumption that the output domain model can be represented in an object-centered representation [3]. In [18], an algorithm was presented to learn action models and a Hierarchical Task Network (HTN) model simultaneously. In [19], an algorithm called LAMP is presented to learn complex action models with quantifiers and logical implications. Despite these successes in action model learning, most previous works only focused on learning action models for *single agents*, and few work addressed the issue for

multi-agent environments. In contrast, to the best of our knowledge, our system LAMMAS is aimed at learning action models for *multi-agent environments* for the first time.

2.3 Multi-Agent Learning

There has been much related work in multi-agent learning. In early work, Guestrin *et al.* (2001) proposed a principled and efficient planning algorithm for cooperative multi-agent dynamic environments [8]. A feature of this algorithm is that the coordination and communication between the agents is not imposed, but derived directly from the system dynamics and function approximation architecture. Bowling (2005) presented a learning algorithm for normal-form games in multi-agent environment. He proved that the algorithm is guaranteed to converge at most zero-average regret, while demonstrating the algorithm converges in many situations of self-play. Wilkinson *et al.* (2005) developed a system to learn an appropriate representation for planning using only an agent's observations and actions [15]. The approach solved two problems, namely, learning an appropriate state-space representation and learning the effects of agent's actions. It required a high-dimensional data set, such as sequences of images, to be given as input. Zhang and Lesser (2010) presented a new algorithm that augmented a basic gradient-ascent algorithm with policy prediction [17]. The key idea behind this algorithm is that a player adjusts its strategy in response to forecasted strategies of the other players, instead of their current ones. None of these algorithms, however, can learn action models for multi-agent planning.

3. PRELIMINARIES

3.1 Satisfiability Problems

The satisfiability problem (SAT) is a decision problem in which, given a propositional logic formula, an assignment of *true* and *false* values are to be determined for the variables to make the entire propositional logic formula true. SAT is known to be NP-Complete [5], but the flip side is that it is very powerful in its representational ability: any propositional logic formula can be re-written as a CNF formula. A CNF formula f is a conjunction of clauses. A clause is a disjunction of literals. A literal l_i is a variable x_i or its negation $\neg x_i$. A variable x_i may take values 0 (for false) or 1 (for true). The length of a clause is the number of its literals. The size of f , denoted by $|f|$, is the sum of the length of all its clauses. An assignment of truth values to the variables satisfies a literal x_i if x_i takes the value 1, satisfies a literal $\neg x_i$ if x_i takes the value 0, satisfies a clause if it satisfies at least one literal of the clause, and satisfies a CNF formula if it satisfies all the clauses of the formula. An empty clause, denoted by \square , contains no literals and cannot be satisfied. An assignment for a CNF formula f is complete if all the variables occurring in f have been assigned; otherwise, it is partial.

The Max-SAT problem for a CNF formula f is the problem of finding an assignment of values to variables that minimizes the number of unsatisfied clauses; equivalently, the aim is to maximize the number of satisfied clauses. There are many SAT solvers for Max-SAT problems, e.g., Maxsatz [10]. In this paper, we use a weighted version of Maxsatz¹ in our LAMMAS system.

3.2 Multi-Agent Planning

Our learning problem is to acquire action models for cooperative Multi-Agent (MA) planning systems, in which agents act under complete state information, and actions have deterministic outcomes. Specifically, we consider problems expressible in a MA-

¹<http://www.laria.u-picardie.fr/~cli/EnglishPage.html>

Table 2: An output example

action models		
<i>hoist</i>	<i>truck</i>	<i>airplane</i>
(:action lift :parameters (?h - hoist ?p - package : ?l - place) :precondition (and (available ?h) (at ?h ?l) : (at ?p ?l)) :effect (and (not (at ?p ?loc)) : (not (available ?h))(lifting ?h ?p))) ... (other action models omitted)	(:action drive :parameters (?t - truck ?from - place : ?to - place ?c - city) :precondition (and (at ?t ?from) (in-city : ?from ?c) (in-city ?to ?c)) :effect (and (not (at ?t ?from)) (at ?t ?to)))	(:action fly :parameters (?a - airplane ?from - airport : ?to - airport) :precondition (at ?a ?from) :effect (and (not (at ?a ?from)) (at ?a ?to)))

Table 1: An input example

		plan trace 1	plan trace 2
plan traces		s_0 (lift hoist1 pkg1 loc1) (load hoist1 pkg1 truck1 loc1) (drive truck1 loc1 airport1 city1) (move hoist1 loc1 airport1 city1) (unload hoist1 pkg1 truck1 airport1) (load hoist1 pkg1 airplane1 airport1) (fly airplane1 airport1 airport2)	...
		g	
predicates		(in-city ?l - place ?c - city) (at ?o - physobj ?l - place) (in ?p - package ?v - vehicle) (lifting ?h - hoist ?p - package) (available ?h - hoist)	
action headings	<i>hoist</i>	(lift ?h - hoist ?p - package ?l - place) (drop ?h - hoist ?p - package ?l - place) (unload ?h - hoist ?p - package ?v - vehicle ?l - place) (load ?h - hoist ?p - package ?v - vehicle ?l - place) (move ?h - hoist ?from - place ?to - place ?c - city)	
	<i>truck</i>	(drive ?t - truck ?from - place ?to - place ?c - city)	
	<i>airplane</i>	(fly ?a - airplane ?from - airport ?to - airport)	

s_0 : (in-city loc1 city1) (in-city airport1 city1) (in-city loc2 city2)
 (in-city airport2 city2) (at plane1 airport1) (at truck1 loc1) (at pkg1 loc1)
 (at hoist1 loc1) (available hoist1)
 g : (at pkg1 airport2) (at plane1 airport2)

extension of the STRIPS language known as MA-STRIPS [4]. Formally, a MA-STRIPS planning problem for a system of agents $\Phi = \{\phi_i\}_{i=1}^k$ is given by a quadruple $\Pi = \langle P, \{\mathcal{A}_i\}_{i=1}^k, s_0, g \rangle$ [2], where:

- P is a finite set of atoms (also called propositions), $s_0 \subseteq P$ encodes the initial situation, and $g \subseteq P$ encodes the goal conditions,
- For $1 \leq i \leq k$, \mathcal{A}_i is the set of action models that the agent ϕ_i is capable of performing. Each action model $a \in \mathcal{A} = \bigcup \mathcal{A}_i$ has the standard STRIPS syntax and semantics, that is, $a = \langle heading(a), pre(a), add(a), del(a) \rangle$, where $heading(a)$ is composed of an action name with zero or more parameters, $pre(a)$, $add(a)$ and $del(a)$ are lists of preconditions, adding effects and deleting effects, respectively.

A solution to an MA-STRIPS problem is a *plan* which is composed of a sequence of ordered actions $\langle a_1, \dots, a_m \rangle$. These actions are executed by different agents to project an initial state s_0 to a goal g . A *plan trace* T is composed of an initial state s_0 , a goal g , partially observed states s_i , and a plan $\langle a_1, \dots, a_m \rangle$ that projects the initial state to the goal, i.e., $T = \{s_0, a_1, s_1, \dots, a_m, g\}$, where the partially observed state s_i can be empty.

3.3 Learning Problem

We formalize our multi-agent learning problem as follows: given a set of plan traces \mathcal{T} , a set of predicates P , and a set of action headings A_i for each agent ϕ_i , Lammas outputs a set of action models \mathcal{A}_i for each agent ϕ_i . We show an input/output example in Tables 1 and 2. The example is taken from the *logistics* domain², extended with the MA-STRIPS conventions ($\langle ?string \rangle$ indicates that $\langle string \rangle$ is a variable). In Table 1 we show an example of *plan trace 1*, likewise for other plan traces. s_0 and g in plan trace 1 are the initial state and the goal, respectively. We assume that there are three agents *hoist*, *truck*, and *airplane*, each of which has its own actions. Agent *hoist* has five actions *lift*, *drop*, *unload*, *load* and *move*, while agents *truck* and *airplane* both have one action, *drive* and *fly* respectively. Note that each parameter of the actions or predicates is associated with a *type*. A *type* can be *primitive*, or composed of other types. In Table 1, the type *physobj* is composed of the types *package*, *hoist*, and *vehicle*; *vehicle* is composed of *truck* and *airplane*; and *place* is composed of *location* and *airport*. Other types *hoist*, *package*, *truck*, *airplane*, *location*, *airport* and *city* are all primitive. In Table 2, we show an example action model for each agent that is learned by our algorithm.

4. THE LAMMAS ALGORITHM

In a nutshell, our Lammas algorithm performs three steps: (1) generate constraints based on the inputs, (2) solve these constraints using a weighted MAX-SAT solver, and (3) extract action models from the solutions. An overview of the Lammas algorithm can be found in Algorithm 1. In the following subsections, we will give a detailed description of each step of Algorithm 1 in turn.

Algorithm 1 An Overview of Our Lammas Algorithm

Input: (1) a set of plan traces \mathcal{T} ; (2) a set of predicates \mathcal{P} ; (3) action headings for each agent ϕ_i : $A_i, i = 1, \dots, n$.

Output: action models for each agent ϕ_i : $\mathcal{A}_i, i = 1, \dots, n$.

- 1: build agent constraints;
- 2: build correctness constraints;
- 3: build action constraints;
- 4: solve all the constraints using a weighted MAX-SAT solver;
- 5: convert the solving result into action models $\mathcal{A}_i, i = 1, \dots, n$;

4.1 Agent Constraints

The first type of constraints is the coordination constraints among different agents (see step 1 of Algorithm 1). With these constraints, we aim at encoding the interactions between the multiple agents, where one agent provides a condition that another agent needs.

²<http://www.cs.toronto.edu/aips2000/>

Specifically, there may be two kinds of actions that any one agent can perform: *interactive* and *non-interactive*. The former requires conditions from other agents or provides conditions for other agents. For example, in plan trace 1 of Table 1, the action “(drive truck1 loc1 airport1 city1)” of agent *truck1* provides the condition “(at truck1 airport1)” for the action “(unload hoist1 pkg1 truck1 airport1)” of agent *hoist1*. Non-interactive actions have no interaction with other agents. For example, in plan trace 1, the action “(lift hoist1 pkg1 loc1)” of agent *hoist1* does not affect other agents or is affected by other agents.³ *Agent constraints* encode constraints for *interactive* actions.

To generate agent constraints, we first collect the set of all possible conditions $PC_i(a)$ for each action a of agent ϕ_i by checking that the proposition’ parameters are included in the action’s, i.e.,

$$PC_i(a) = \{p \mid para(p) \subseteq para(a), \text{ for each } p \in P\},$$

where $para(p)$ denotes the set of parameters of p , likewise for $para(a)$.

Example 1: In Table 1, let ϕ_1, ϕ_2, ϕ_3 be agents *hoist*, *truck*, *airplane*, respectively. We can build possible conditions for each agent’s actions as follows:⁴

$$\begin{aligned} PC_1(lift) &= \{(at ?h ?l), (at ?p ?l), (lifting ?h ?p), \\ &\quad (available ?h)\}; \\ PC_1(drop) &= \{(at ?h ?l), (at ?p ?l), (lifting ?h ?p), \\ &\quad (available ?h)\}; \\ PC_1(unload) &= \{(at ?h ?l), (at ?p ?l), (at ?v ?l), (in ?p ?v), \\ &\quad (lifting ?h ?p), (available ?h)\}; \\ PC_1(load) &= \{(at ?h ?l), (at ?p ?l), (at ?v ?l), (in ?p ?v), \\ &\quad (lifting ?h ?p), (available ?h)\}; \\ PC_1(move) &= \{(at ?h ?from), (at ?h ?to), (in-city ?from ?c), \\ &\quad (in-city ?to ?c)\}; \\ PC_2(drive) &= \{(at ?t ?from), (at ?t ?to), (in-city ?from ?c), \\ &\quad (in-city ?to ?c)\}; \\ PC_3(fly) &= \{(at ?a ?from), (at ?a ?to)\}. \end{aligned}$$

After collecting all possible conditions, we compute all common conditions among pairs of actions (a, a') from agent pairs (ϕ_i, ϕ_j) . To do this, we identify an one-to-one correspondence from a subset of parameters of a to a subset of parameters of a' such that the parameter m and its corresponding parameter m' can be instantiated with the same value. Two parameters can be instantiated with the same value if (1) they have the same type or (2) one is a subtype of the other one (i.e., truck is a type of vehicle). We denote any one such one-to-one correspondence as $CR_{a,a'}$ and the corresponding parameters in $CR_{a,a'}$ as pairs (m, m') , where m and m' are the indexes of parameters of a and a' . We denote the common conditions for a pair of actions (a, a') of two agents ϕ_i and ϕ_j relative to a correspondence $CR_{a,a'}$ as $PC_{ij}(a, a', CR_{a,a'})$. For example, take $PC_2(drive)$ and $PC_1(unload)$. Assuming

$$CR_{drive,unload} = \{(1, 3), (3, 4)\}$$

(i.e., the first parameter of *drive* corresponds to the third parameter of *unload*, and the third parameter of *drive* corresponds to the

³We consider *hoist1* and *hoist2* as instances of the same agent because they share the same action models.

⁴For simplicity, we omit the *type* associated with each parameter of each predicate (e.g., type “package” of parameter “?p” is omitted).

fourth parameter of *unload*), we have

$$PC_{21}(drive, unload, CR_{drive,unload}) = \{(at ?t ?to)\}$$

or

$$PC_{21}(drive, unload, CR_{drive,unload}) = \{(at ?v ?l)\}.$$

We say that an action a of an agent ϕ_i is *interactive* with another action a' of agent ϕ_j , if and only if there exists a correspondence $CR_{a,a'}$ such that $PC_{ij}(a, a', CR_{a,a'})$ is not empty. Otherwise, we say that action a is *non-interactive* with action a' , and we say that action a is *noninteractive* if it is non-interactive with all the actions of other agents. For example, in Example 1, action *drive* is interactive with action *unload*, while action *lift* of agent *hoist1* is non-interactive.

In the next step, we generate agent constraints by finding all the *interactive* actions among agents and building a new structure that we call a **weighted Agent Interaction Graph** (w -AIG). We do this by scanning all the plan traces. We define a w -AIG by a tuple (N, E, W) , where N is a set of nodes, E is a set of edges, and W is a set of weights. The nodes N correspond to agents in Φ . A directed edge in E from an agent ϕ_i to another agent ϕ_j is labeled by $PC_{ij}(a, a', CR_{a,a'})$, indicating that actions $a \in \mathcal{A}_i$ and $a' \in \mathcal{A}_j$ satisfy

$$(Add(a) \cap Pre(a')) \subseteq PC_{ij}(a, a', CR_{a,a'}).$$

It is possible that there are multiple edges between the same two agents.

Each weight in W is associated with an edge in E , measuring the likelihood of the existence of that edge. This likelihood is computed as follows. We scan the set of plan traces \mathcal{T} , looking for situations in which ϕ_j executes actions immediately after ϕ_i . In such situations, we conjecture that some actions of agent ϕ_i in plan traces probably provides some conditions for some actions of agent ϕ_j , which corresponds to some edges from ϕ_i to ϕ_j in w -AIG. The same edges may be repeatedly created when scanning plan traces. Each time the same edge is found, its corresponding weight will be incremented by one. The procedure for building the graph is shown in Algorithm 2.

In step 4 of Algorithm 2, $lengthof(t)$ returns the number of actions in t . In step 8, $findCR(a_k, a_{k'})$ returns a set of corresponding parameters between a and a' . For example, let a_k and $a_{k'}$ be actions “(drive truck1 loc1 airport1 city1)” and “(unload hoist1 pkg1 truck1 airport1)”. The first parameter “truck1” of a_k is the same as the third parameter of $a_{k'}$, and the third parameter “airport1” of a_k is the same as the fourth parameter of $a_{k'}$. Thus, the procedure $findCR(a_k, a_{k'})$ returns $\{(1, 3), (3, 4)\}$ as the corresponding parameters between a and a' . In step 13, $W(e)$ records the times that edge e is repeatedly found.

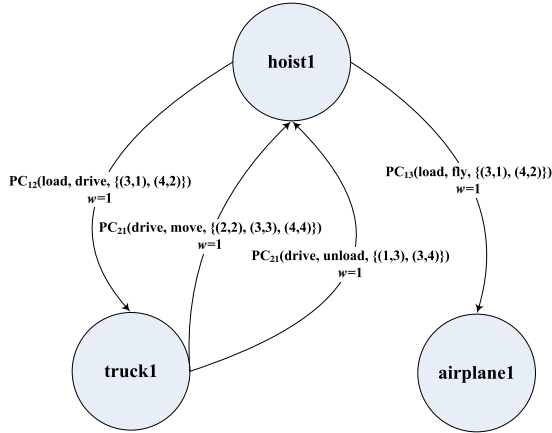
Example 2: From Example 1, we can easily build a w -AIG after scanning plan trace 1, as shown in Figure 1. After scanning plan trace 1, we know that the first two actions *lift* and *load* are executed by agent *hoist1*, and the third action *drive* is executed by agent *truck1*. Since that *lift* is *noninteractive* holds, it is not included. For action *load*, since $PC_{12}(load, drive, \{(3, 1), (4, 2)\}) \neq \emptyset$, a new edge e is created, and its weight is set as one. Likewise, we can create other edges by scanning plan trace 1.

Once the w -AIG is generated, the last step is to generate agent constraints. Let e be an edge that connects an agent ϕ_i to another agent ϕ_j . We can build the constraints to denote that some action of agent ϕ_i provides some condition for some action of agent ϕ_j . Formally, for each edge connecting agent ϕ_i to agent ϕ_j with a label $PC_{ij}(a, a', CR_{a,a'})$, we create the following constraints (one

Algorithm 2 Building w -AIG: $G = buildwAIG(\mathcal{T})$

input: a set of plan traces \mathcal{T} .**output:** a w -AIG $G = (N, E, W)$.

```
1: let  $N = \Phi, E = \emptyset$ ;
2: for each  $t \in \mathcal{T}$  do
3:    $n = 1$ ;
4:   while  $n \leq \text{lengthof}(t)$  do
5:     find the maximal number  $h$ , such that actions
      $a_n, \dots, a_{n+h}$  in  $t$  are all executed by agent  $\phi_i$ ;
6:     find the maximal number  $h'$ , such that actions
      $a_{n+h+1}, \dots, a_{n+h+h'}$  in  $t$  are executed by agent  $\phi_j$ ;
7:     for each two integers  $k \in [n, n+h]$  and  $k' \in [n+h+1, n+h+h']$  do
8:       assuming  $a_k$  and  $a_{k'}$  are instances of action  $a$  and  $a'$ 
       respectively, build  $CR_{a,a'} = \text{findCR}(a_k, a_{k'})$ ;
9:       calculate  $PC_{ij}(a, a', CR_{a,a'})$ ;
10:      if  $PC_{ij}(a, a', CR_{a,a'}) \neq \emptyset$  then
11:        create an edge  $e$ , with label  $PC_{ij}(a, a', CR_{a,a'})$ ;
12:        if  $e \in E$  then
13:           $W(e) = W(e) + 1$ ;
14:        else
15:           $E = E \cup \{e\}$ , and  $W(e) = 1$ ;
16:        end if
17:      end if
18:    end for
19:     $n = n + h + 1$ ;
20:  end while
21: end for
22: return  $(N, E, W)$ ;
```

**Figure 1: An example of w -AIG**for each $p \in PC_{ij}(a, a', CR_{a,a'})$:

$$p \in \text{Add}_i(a) \wedge p \in \text{Pre}_j(a').$$

The weights of these constraints are directly assigned by the values of W .

4.2 Correctness Constraints

In step 2 of Algorithm 1, we build *correctness constraints* (first introduced by [16]), where we require that the action models learned are consistent with the training plan traces. These constraints are imposed on the relationship between ordered actions in the plan traces to ensure that the causal links in the plan traces are not broken. That is, for each precondition p of an action a_j in a plan trace,

either p is in the initial state, or there is an action a_i ($i < j$) prior to a_j that adds p and there is no action a_k ($i < k < j$) between a_i and a_j that deletes p . For each literal q in a state s_j , either q is in the initial state s_0 , or there is an action a_i before s_j that adds q while no action a_k deletes q .

We formulate these constraints as follows.

$$p \in \text{Pre}(a_j) \wedge p \in \text{Add}(a_i) \wedge p \notin \text{Del}(a_k)$$

and

$$q \in g \wedge (q \in s_0 \vee (q \in \text{Add}(a_i) \wedge q \notin \text{Del}(a_k)))$$

where $i < k < j$, $\text{Del}(a_j)$ is a set of deleting predicates of the action a_j and g is the goal which is composed of a set of propositions.In order to ensure that the correctness constraints are maximally satisfied, we assign these constraints with a maximal weight among all weights W in w -AIG.

4.3 Action Constraints

In step 3 of Algorithm 1, we build another kind of constraint known as *action constraints* (introduced by [16]). We introduce two categories of action constraints. The first is the result of the semantics of STRIPS [4], while the second is the result of the statistical information extracted from the plan traces (i.e., the relationship between states and actions revealed by plan traces). Specifically, we build the constraints as follows.

1. In STRIPS, if a predicate p is a precondition of an action a , i.e., $p \in \text{pre}(a)$, then it should not be added by a , i.e., $p \notin \text{Add}(a)$; on the other hand, if a predicate q is added by an action a , i.e., $q \in \text{Add}(a)$, then it should not be deleted by a at the same time, i.e., $q \notin \text{Del}(a)$. Formally, these constraints can be represented by

$$(p \in \text{Pre}(a) \rightarrow p \notin \text{Add}(a))$$

and

$$(q \in \text{Add}(a) \rightarrow q \notin \text{Del}(a))$$

for any action a from any agent. The weights of these constraints are also set as the maximal value of W in w -AIG.

2. In general, if a predicate p frequently occurs before an action a in plan traces (i.e., p frequently occurs in the state where a is executed), then p is likely a precondition of a . Similarly, if a predicate q frequently occurs after a (i.e., q frequently occurs in the state after a is executed), then q is likely an added effect of a . This idea can be formulated via the following constraints:

$$(p \in \text{before}(a) \rightarrow p \in \text{Pre}(a))$$

and

$$(q \in \text{after}(a) \rightarrow q \in \text{Add}(a))$$

where $\text{before}(a)$ indicates a set of predicates that occur frequently before a , while $\text{after}(a)$ indicates a set of predicates that occur frequently after a , where the term *frequently* is used to indicate that the number of occurrences is larger than a pre-defined threshold; in each domain we need to adjust the threshold value empirically. The weights of these constraints are set as the number of their occurrences.

4.4 Attaining Action Models

After all three types of constraints are built, we satisfy all constraints using a weighted MAX-SAT solver (Step 4 of Algorithm 1). Before that, we introduce three new parameters λ_i ($1 \leq i \leq 3$) to control the relative importance of the three kinds of constraints (which is similar to [18]). We adjust the weights of each kind of constraints by replacing their weights with $\frac{\lambda_i}{1-\lambda_i}w_i$, where w_i is the weight of the i th kind of constraints. By adjusting λ_i from 0 to 1, we can adjust the weight from 0 to ∞ . The weighted MAX-SAT solution returns a truth value for each atom. We are interested specifically in the truth values of atoms of the form “ $p \in pre(a)$ ”, “ $p \in add(a)$ ”, and “ $p \in del(a)$ ”. We convert the MAX-SAT solution to action models directly: if an atom “ $p \in Add(a)$ ” is assigned with *true*, p will be converted to an adding effect of action a . We can likewise transform an atom to a precondition or a negative effect of an action.

5. EXPERIMENTS

In order to verify the effectiveness of `Lammas`, we developed a prototype system, which we compare to a baseline algorithm on three multi-agent domains derived from IPC (International Planning Competition) domains. These domains are multi-agent variations of *logistics*⁵, *rovers*⁶ and *openstacks*⁷.

5.1 Dataset and Criterion

In the first domain *logistics*, a set of packages should be moved on a roadmap from their initial to their target locations using the given vehicles. The packages can be loaded onto and unloaded off the vehicles, and each vehicle can move along a certain subset of road segments. We extend this domain by introducing three kinds of agents *hoist*, *truck* and *airplane*, each of which has its own actions, as it is described in Table 1. These agents cooperate to achieve the specific goals, e.g., a *hoist* agent uploads a package to a truck in its starting location; a *truck* agent takes the package from this location to an airport, a *hoist* agent then unloads the package from the truck and loads it into an airplane, an *airplane* agent takes the package from an airport to another airport and so forth. The second domain is *rovers*, which is inspired by a planetary rovers planning problem. The domain *rovers* tasks a collection of rovers with navigating a planetary surface, finding samples (soil, rock and image), analyzing them, and communicating the results back to a lander. We extend this domain by introducing four kinds of agents: *soilrover*, *rockrover*, *imagerover* and *communicant*, each of which has its own actions. Specifically, the agents *soilrover*, *rockrover* and *imagerover* perform actions related to sampling soil, rocks, and images, respectively. In other words, they are responsible for transporting the soil to agent *communicant*. The agent *communicant* is in charge of analyzing the soil, rocks and images, and communicating the results back to a lander. In the last domain *openstacks*, a manufacturer has a number of orders, each consisting of a combination of different products, and can only make one product at a time. We extend this domain by introducing three kinds of agents: *receiver*, *producer* and *deliveryman*. Agent *receiver* receives orders from clients and passes them to producers. Agent *producer* produces products according to the orders and passes them to delivery men. Agent *deliveryman* delivers products to clients according to the orders. With these extended domains, we can test our learning algorithm in multi-agent conventions. In what follows, we refer to these extended multi-agent domains as *ma-logistics*, *ma-rovers* and

ma-openstacks, respectively. The action models of these extended domains were built by hand and used as *ground truth* action models. Using the *ground truth* action models, we generated 200 plan traces from each domain, which was used as the training data for `Lammas`.

We compare the learned action models with the ground truth action models to calculate the error rates. If a precondition appears in the precondition list of our learned action model but not in the precondition list of its corresponding ground-truth action model, the error count of preconditions, denoted by E_{pre} , is incremented by one (this is a false positive). If a precondition appears in the precondition list of a ground truth action model but not in the precondition list of the corresponding learned action model, E_{pre} also is incremented by one (this is a false negative). Likewise, the error count in the actions’ adding (or deleting) lists is denoted by E_{add} (or E_{del}). False positives restrict the potential plans that could be generated, and they measure the loss in terms of the completeness of planning. False negatives can give rise to incorrect plans, and thus they measure the loss in the soundness.

We use T_{pre} , T_{add} and T_{del} to denote the number of all the possible preconditions, add-effects and delete-effects of an action model, respectively. We define the error rate of an action model a as

$$R(a) = \frac{1}{3} \left(\frac{E_{pre}}{T_{pre}} + \frac{E_{add}}{T_{add}} + \frac{E_{del}}{T_{del}} \right)$$

where we assume the error rates of preconditions, adding effects and deleting effects are equally important, and the range of error rate $R(a)$ is within $[0,1]$. Furthermore, we define the error rate of all the action models from agents Φ in a domain as

$$R(\Phi) = \frac{1}{\sum_{i \in \Phi} |\mathcal{A}_i|} \sum_{i \in \Phi} \sum_{a \in \mathcal{A}_i} R(a)$$

where, $|\mathcal{A}_i|$ is the number of action models that the agent ϕ_i is capable of performing. Using this definition of error rate as the performance metric, we present our experimental results in the next subsection.

5.2 Experimental Results

We test our `Lammas` algorithm in the following way. First, we compare between our `Lammas` algorithm and ARMS. Second, we vary the weights of each type of constraints and observe the impact on performance. Finally, we report on the running time of `Lammas`.

5.2.1 Comparison between `Lammas` and ARMS

One way to conduct the comparison is to consider the existence of an *oracle agent*, which knows all the actions of each agent in the multi-agent system, such that the multi-agent system can be viewed as a single-agent system. This will enable the learning of actions for the oracle agent by using previous single-agent action-model learning algorithms such as ARMS. These single-agent learning algorithms, however, do not consider the coordination information involved in the various agents. In `Lammas` this information is captured by the agent constraints. We hypothesize that the `Lammas` algorithm can handle these interactions better, resulting in reduced error rates.

We set the percentage of observed states as $1/5$, which indicates that one in five consecutive states can be observed. We set the percentage of observed propositions in each observed state as $1/5$. We also set all λ_i ($1 \leq i \leq 3$) as 0.5 without any bias. We ran `Lammas` and ARMS five times by randomly selecting the states and propositions in plan traces, and calculate the average of error rates. The comparison result is shown in Figure 2.

⁵<http://www.cs.toronto.edu/aips2000/>

⁶<http://planning.cis.strath.ac.uk/competition/>

⁷<http://zeus.ing.unibs.it/ipc-5/>

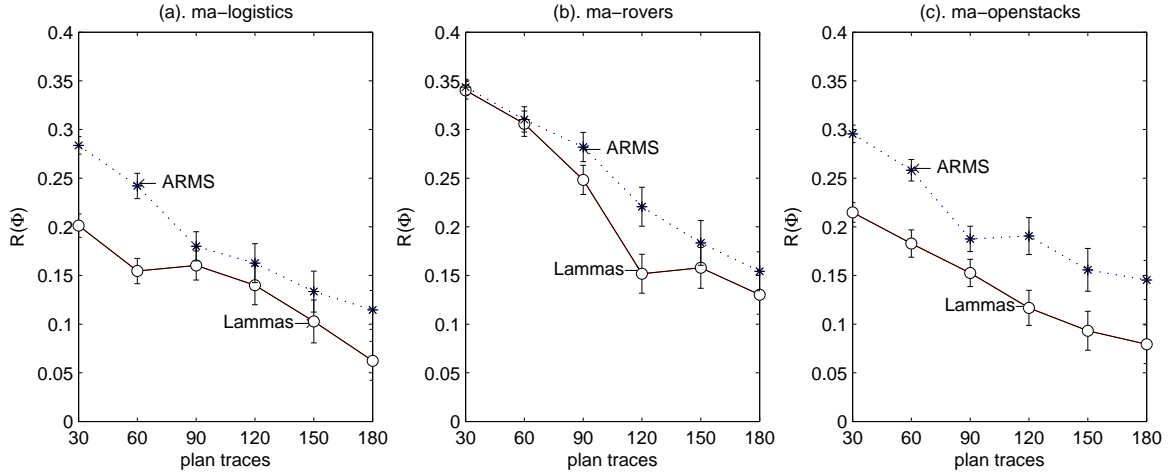


Figure 2: The comparison between Lammas and ARMS

From Figure 2, we can see that the error rates $R(\Phi)$ of both Lammas and ARMS generally decrease as the number of plan traces increases, which is consistent with the intuition that when more training data is given the percentage of error will decrease. By observation, we can also find that the error rate $R(\Phi)$ of our Lammas algorithm is generally smaller than that of ARMS, which suggests that the agent constraints generated from w -AIG can indeed help improve the learning result in multi-agent environments. From the curves for different domains, our Lammas algorithm functions much better in both domains of *ma-logistics* and *ma-openstacks* than in the domain *ma-rovers*. The results are statistically significant; we performed the Student’s t-test and the results are 0.0101 for *ma-logistics*, 0.0416 for *ma-rovers* and 0.0002 for *ma-openstacks*. This suggests that the agent constraints work better in *ma-logistics* and *ma-openstacks* than in *ma-rovers*. The reason for this difference is because agents in *ma-logistics* and *ma-openstacks* have more interactions between each other than that in *ma-rovers*.

5.2.2 Varying weights of constraints

The importance of different kinds of constraints may be different in the learning process. We test this hypothesis by varying the weights of the different kinds of constraints. We fix λ_2 and λ_3 as 0.5 and set λ_1 as different values of 0, 0.25, 0.5, 0.75 and 1. We run Lammas and calculate the error rates with respect to different values of λ_1 . The error rates are shown in the second/fifth/eighth columns of Table 3. Likewise, we calculate the error rates with different values of λ_2 or λ_3 when fixing the other two λ values at 0.5, as shown in Table 3. In the table, we highlight the smallest error rates of each column with boldface; e.g., in the second column the smallest error rate is 0.0601 where $\lambda_1 = 0.75$.

From Table 3, we find that λ_i cannot be set too high (the highest being 1) or set too low (the lowest being 0); otherwise its corresponding error rates will be high. This suggests that the weights of constraints cannot be set too high or too low to offset the impact of other constraints. Hence, all three kinds of constraints are needed for learning high quality result. For instance, when λ_i is set too high, its corresponding kind of constraints plays a major role while the other two kinds of constraints play a relatively minor role (in an extreme case, they play no effect when $\lambda_i = 1$) on the learning result. On the other hand, when λ_i is set too low, the importance of its corresponding kind of constraints is reduced. In the extreme case, they have no effect when $\lambda_i = 0$.

By comparing the λ_1 columns between *ma-logistics* and *ma-rovers*, we can see that the value of λ_1 should be higher in *ma-logistics* (to make error rates smaller) than in *ma-rovers*, which suggests agent constraints in *ma-logistics* are more important than in *ma-rovers*. The reason for this is because there are more interactions among agents in *ma-logistics* than in *ma-rovers*. Hence, exploiting the agent’s interaction information helps improve the learning result.

5.2.3 Running time

To test the running time of the Lammas algorithm, we set $\lambda_i (1 \leq i \leq 3)$ as 0.5 and run Lammas with respect to different number of plan traces. The result is shown in Figure 3. As can be seen from the figure, the running time increases polynomially with the number of input plan traces. This can be verified by fitting the relationship between the number of plan traces and the running time to a performance curve with a polynomial of order 2 or 3. For example, the fit polynomial for *ma-logistics* is $-0.0002x^3 + 0.0541x^2 - 3.1616x + 52.6667$.

ARMS also runs in polynomial time on the size of the input traces. Hence, since both ARMS and Lammas are designed to run off-line there is no real advantage of using one or the other based on their running times. However, our experiments show that Lammas has the advantage that it can learn more accurate models than ARMS for multi-agent environments.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented an action-model learning system known as Lammas, which performs well in multi-agent domains. We learn the structure information to reflect agent interactions, which is shown empirically to improve the quality of the learned action models. Our approach builds a w -AIG graph to reveal the potential interactions among agents, which results in agent constraints that are used to capture agent interactions. Integrating these agent constraints with previously used action constraints are shown to give better learning performance. Our experiments show that Lammas is effective in three benchmark domains.

Our work can be extended to more complex multi-agent domains. For example, in a multi-player computer game setting, agents have their own utilities, and they may cooperate with each other or work against each other. In such situations, we may incorporate more types of constraints to model adversarial situations.

Table 3: Error rates with respect to different λ values

λ_i values	<i>ma-logistics</i>			<i>ma-rovers</i>			<i>ma-openstacks</i>		
	λ_1	λ_2	λ_3	λ_1	λ_2	λ_3	λ_1	λ_2	λ_3
1	0.1552	0.1784	0.1744	0.1305	0.1714	0.1552	0.1202	0.1323	0.1544
0.75	0.0601	0.2020	0.1561	0.1329	0.1081	0.1271	0.0986	0.0633	0.1561
0.5	0.0623	0.0623	0.0623	0.1302	0.1302	0.1302	0.0794	0.0794	0.0794
0.25	0.0943	0.1436	0.1594	0.1164	0.1490	0.0962	0.1118	0.1561	0.1294
0	0.1236	0.1934	0.2479	0.1610	0.1648	0.2124	0.1638	0.2134	0.1979

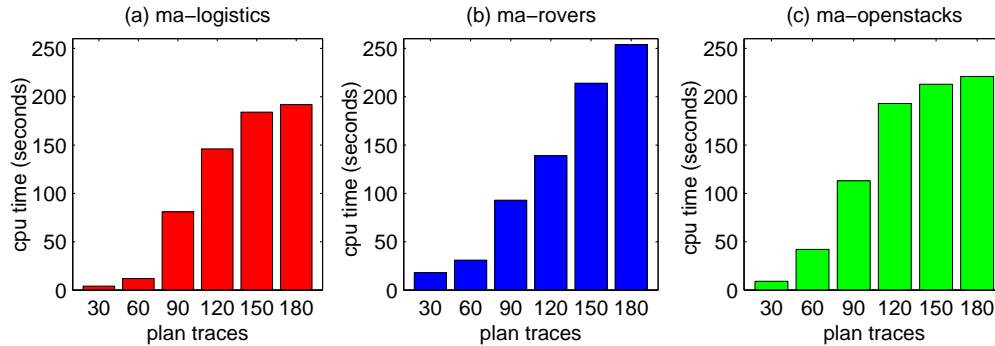


Figure 3: The running time of the Lammas algorithm

Acknowledgement

Hankui Zhuo thanks China Postdoctoral Science Foundation funded project(Grant No.20100480806) and National Natural Science Foundation of China (61033010) for support of this research. Qiang Yang thanks Hong Kong RGC/NSFC grant N HKUST624/09 for support of this research. Hector Munoz-Avila thanks the National Science Foundation grant 0642882 for support of this research.

7. REFERENCES

- [1] E. Amir. Learning partially observable deterministic action models. In *Proceedings of IJCAI'05*, 2005.
- [2] R. I. Brafman and C. Domshlak. From one to many: Planning for loosely coupled multi-agent systems. In *Proceedings of ICAPS'08*, 2008.
- [3] S. Cresswell, T. L. McCluskey, and M. M. West. Acquisition of object-centred domain models from planning examples. In *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling (ICAPS'09)*, 2009.
- [4] R. Fikes and N. J. Nilsson. STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence Journal*, pages 189–208, 1971.
- [5] M. R. Garey and D. S. Johnson. *Computers and intractability: A guide to the theory of np-completeness*. W.H. Freeman, 1979.
- [6] M. Georgeff. A theory of action for multiagent planning. In *Proceedings of AAAI'84*, 1984.
- [7] Y. Gil. Learning by experimentation: Incremental refinement of incomplete planning domains. In *Proceedings of the Eleventh International Conference on Machine Learning (ICML-94)*, pages 87–95, 1994.
- [8] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored mdps. In *Proceedings of NIPS'01*, 2001.
- [9] M. P. Holmes and C. L. Isbell, Jr. Schema learning: Experience-based construction of predictive action models. In *In Advances in Neural Information Processing Systems 17 (NIPS-04)*, 2004.
- [10] C. M. LI, F. Manyá, and J. Planes. New inference rules for Max-SAT. *Journal of Artificial Intelligence Research*, 30:321–359, October 2007.
- [11] R. Nissim, R. I. Brafman, and C. Domshlak. A general, fully distributed multi-agent planning algorithm. In *Proceedings of AAMAS'10*, 2010.
- [12] T. J. Walsh and M. L. Littman. Efficient learning of action schemas and web-service descriptions. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI-08)*, pages 714–719, 2008.
- [13] X. Wang. Learning by observation and practice: An incremental approach for planning operator acquisition. In *Proceedings of the Twelfth International Conference on Machine Learning (ICML-95)*, pages 549–557, 1995.
- [14] D. E. Wilkins and K. L. Myers. A multiagent planning architecture. In *Proceedings of AIPS'98*, 1998.
- [15] D. Wilkinson, M. Bowling, and A. Ghodsi. Learning subjective representations for planning. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 889–894, 2005.
- [16] Q. Yang, K. Wu, and Y. Jiang. Learning action models from plan examples using weighted MAX-SAT. *Artificial Intelligence Journal*, 171:107–143, February 2007.
- [17] C. Zhang and V. Lesser. Multi-Agent Learning with Policy Prediction. In *Proceedings of the 24th National Conference on Artificial Intelligence (AAAI'10)*, Atlanta, GA, USA, 2010.
- [18] H. H. Zhuo, D. H. Hu, C. Hogg, Q. Yang, and H. Muñoz-Avila. Learning HTN method preconditions and action models from partial observations. In *Proceedings of IJCAI*, pages 1804–1810, 2009.
- [19] H. H. Zhuo, Q. Yang, D. H. Hu, and L. Li. Learning complex action models with quantifiers and logical implications. *Artificial Intelligence Journal*, 174(18):1540–1569, 2010.

Theoretical Considerations of Potential-Based Reward Shaping for Multi-Agent Systems

Sam Devlin
University of York, UK

Daniel Kudenko
University of York, UK

ABSTRACT

Potential-based reward shaping has previously been proven to both be equivalent to Q-table initialisation and guarantee policy invariance in single-agent reinforcement learning. The method has since been used in multi-agent reinforcement learning without consideration of whether the theoretical equivalence and guarantees hold. This paper extends the existing proofs to similar results in multi-agent systems, providing the theoretical background to explain the success of previous empirical studies. Specifically, it is proven that the equivalence to Q-table initialisation remains and the Nash Equilibria of the underlying stochastic game are not modified. Furthermore, we demonstrate empirically that potential-based reward shaping affects exploration and, consequentially, can alter the joint policy converged upon.

Categories and Subject Descriptors

I.2.6 [Artificial Intelligence]: Learning; I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*Multiagent Systems*

General Terms

Theory, Experimentation

Keywords

Reinforcement Learning, Reward Shaping, Multiagent Learning, Reward Structures for Learning.

1. INTRODUCTION

Current trends are showing a rise in interest in Multi-Agent Systems (MAS). With multiple, distributed agents a larger set of problem domains can be practically modelled [35]. To control each agent, a reinforcement learning solution can provide adaptive, autonomous, and self-improving agents.

However, whilst reinforcement learning can handle problems with combinatorial state spaces in single-agent problem domains [21, 26], adding more agents to the same environment is a significant challenge [5]. Specifically, as the other

Cite as: Theoretical Considerations of Potential-Based Reward Shaping for Multi-Agent Systems, Sam Devlin and Daniel Kudenko, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 225-232.
Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

agents execute their own actions they subsequently influence the state of the world. This makes the environment appear non-stationary to an individual agent because other agents may concurrently learn and change their behaviour. Unknown to the agent, the same local state-action pair will have a different transition function even though the global state-joint action pair has not changed.

It has been shown in single-agent reinforcement learning that the quicker a learning agent can reach convergence in its policy the more it will benefit from instability in the environment, as it is better suited to adapt to changes [30]. But in MAS the state-action space grows exponentially with the number of agents, which may considerably slow down convergence reducing agents' ability to adapt quickly. Therefore, methods of reducing the time to convergence are of significant importance when implementing reinforcement learning solutions to MAS.

One such method, empirically demonstrated to decrease the time for each individual learning in a common environment to converge on a stable policy, is incorporating heuristic knowledge [17, 25]. However, most existing reinforcement learning algorithms were proposed under the assumption that there is no knowledge available about the problem. This is often not the case; in many practical applications heuristic knowledge can be easily identified by the designer of the system [23], or acquired using reasoning or learning [10].

In single-agent reinforcement learning, potential-based reward shaping has been proven to be a principled and theoretically correct method of incorporating heuristic knowledge into an agent. Provided domain knowledge dependent on states alone, receiving an additional potential-based reward of the correct form does not alter the optimal policy of an agent [20].

To date, applications of potential-based reward shaping to MAS [2, 16] have been studied without published consideration of whether the proofs, originally intended for single-agent problem domains, hold for multi-agent reinforcement learning.

The bulk of our findings, discussed in Section 4, consider the theoretical implications for reward shaping of changing from single-agent problem domains to MAS. This work focuses on the analysis of two fundamental results in single-agent, potential-based reward shaping; the equivalence to Q-table initialisation [33] and the invariance of policies between shaped and non-shaped agents provided [20].

The first remains constant, potential-based reward shaping is equivalent to Q-table initialisation regardless of the

number of agents learning in the environment. The latter, however, takes new meanings in a MAS. The goal of single-agent reinforcement learning is to compute the policy of maximum reward but with multiple agents, potentially competing, the goal becomes Nash Equilibrium [19]. Therefore, the multi-agent equivalent to policy invariance [20], successfully proven in this paper, is that potential-based reward shaping does not alter the Nash Equilibria of the MAS.

However, potential-based reward shaping can have implications for the joint policy a multi-agent reinforcement learning solution will converge to. As we will show, the final joint policy will still be a Nash Equilibrium of the original system (i.e., before any agents received reward shaping) but may not be the same as the additional reward alters the individual agent’s exploration which affects the experiences all agents will have.

We close, in Section 5 by empirically demonstrating our findings but first, to begin, the following section will review existing work and the required background knowledge.

2. EXISTING WORK

2.1 Reinforcement Learning

Reinforcement learning is a paradigm which allows agents to learn by reward and punishment from interactions with the environment [28]. The numeric feedback received from the environment is used to improve the agent’s actions. The majority of work in the area of reinforcement learning applies a Markov Decision Process (MDP) as a mathematical model [22].

An MDP is a tuple $\langle S, A, T, R \rangle$, where S is the state space, A is the action space, $T(s, a, s') = Pr(s'|s, a)$ is the probability that action a in state s will lead to state s' , and $R(s, a, s')$ is the immediate reward r received when action a taken in state s results in a transition to state s' . The problem of solving an MDP is to find a policy (i.e., mapping from states to actions) which maximises the accumulated reward. When the environment dynamics (transition probabilities and a reward function) are available, this task can be solved using iterative approaches like policy and value iteration [3].

When the environment dynamics are not available, as with most true environments, value iteration cannot be used. However, the concept of an iterative approach remains the backbone of the majority of reinforcement learning algorithms. These algorithms apply so called temporal-difference updates to propagate information about values of states, $V(s)$, or state-action, $Q(s, a)$, pairs [27]. These updates are based on the difference of the two temporally different estimates of a particular state or state-action value. The Q-learning algorithm is such a method [28]. After each transition, $(s, a) \rightarrow (s', r)$, in the environment, it updates state-action values by the formula:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (1)$$

where α is the rate of learning and γ is the discount factor. It modifies the value of taking action a in state s , when after executing this action the environment returned reward r , and moved to a new state s' .

2.2 Multi-Agent Reinforcement Learning

Applications of reinforcement learning to MAS typically take one of two approaches; multiple individual learners or joint action learners [6]. The former is the deployment of multiple agents each using a single-agent reinforcement learning algorithm. The latter is a group of multi-agent specific algorithms designed to consider the existence of other agents.

Multiple individual learners assume any other agents to be a part of the environment and so, as the others simultaneously learn, the environment appears to be dynamic as the probability of transition when taking action a in state s changes over time. To overcome the appearance of a dynamic environment, joint action learners were developed that extend their value function to consider for each state the value of each possible combination of actions by all agents.

Learning by joint action, however, breaks a common fundamental concept of MAS in which each agent is self motivated and so may not consent to the broadcasting of their action choices. Furthermore, the consideration of the joint action causes an exponential increase in the number of values that must be calculated with each additional agent added to the system. Typically, joint action learning algorithms have only been demonstrated in trivial problem domains [31, 11, 6] whilst applications in complex systems most often implement multiple individual learners [18, 29, 30]. For these reasons, this work will focus on multiple individual learners and not joint action learners. However, these proofs can be extended to cover joint action learners, those we have specifically considered include MiniMax Q-learning [14], Friend-or-Foe Q-learning [15] and Nash-Q [11].

Unlike single-agent reinforcement learning where the goal is to maximise the individual’s reward, when multiple self motivated agents are deployed not all agents can always receive their maximum reward. Instead some compromise must be made, typically the system is designed aiming to converge to a Nash Equilibrium [24]. Multiple individual learners will, given sufficient learning time, converge to a point of equilibrium, however, no guarantees can be made that this will be the optimum Nash Equilibrium [6].

To model a MAS, the single-agent MDP becomes inadequate and instead the more general Stochastic Game (SG) is required [5]. A SG of n agents is a tuple $\langle S, A_1, \dots, A_n, T, R_1, \dots, R_n \rangle$, where S is the state space, A_i is the action space of agent i , $T(s, A, s') = Pr(s'|s, A)$ is the probability that joint action A in state s will lead to state s' , and $R_i(s, a, s')$ is the immediate reward r received by agent i when action a taken in state s results in a transition to state s' [9].

3. REWARD SHAPING

The immediate reward r , which is in the update rule given by Equation 1, represents the feedback from the environment. The idea of *reward shaping* is to provide an additional reward which will improve the convergence of the learning agent with regard to the learning speed [20, 23]. This concept can be represented by the following formula for the Q-learning algorithm:

$$Q(s, a) \leftarrow Q(s, a) + \alpha[r + F(s, s') + \gamma \max_{a'} Q(s', a') - Q(s, a)] \quad (2)$$

where $F(s, s')$ is the general form of the shaping reward.

Even though reward shaping has been powerful in many

experiments it quickly became apparent that, when used improperly, it can change the optimal policy [23]. To deal with such problems, potential-based reward shaping was proposed [20] as the difference of some potential function Φ defined over a source s and a destination state s' :

$$F(s, s') = \gamma\Phi(s') - \Phi(s) \quad (3)$$

where γ must be the same discount factor as used in the agent's update rule (see Equation 1).

Ng et al. [20] proved that potential-based reward shaping, defined according to Equation 3, guarantees learning a policy which is equivalent to the one learnt without reward shaping in both infinite and finite horizon MDPs.

Wiewiora [33] later proved that an agent learning with potential-based reward shaping and no knowledge-based Q-table initialisation will behave identically to an agent without reward shaping when the latter agent's value function is initialised with the same heuristic knowledge represented by $\Phi(s)$. This is an important fact, because when function approximation is used in big environments, where the structural properties of the state space are not clear, it is not easy to initialise the value function. Potential-based reward shaping represents a flexible and theoretically correct method to incorporate background knowledge regarding states into reinforcement learning algorithms.

3.1 Reward Shaping In Multi-Agent Systems

Incorporating heuristic knowledge has been shown to be beneficial in multi-agent reinforcement learning [2, 16, 17, 25]. However, some of the previous examples did not use potential-based functions to shape the reward [17, 25] and could potentially, therefore, suffer from introducing beneficial cyclic policies that cause convergence to an unintended behaviour as demonstrated previously in a single-agent problem domain [23].

The remaining applications that were potential-based [2, 16], demonstrated an increased probability of convergence to a higher value Nash Equilibrium. As it has long been established that multiple individual learners are not guaranteed to converge to the optimal Nash Equilibrium [6], a number of methods to increase the probability of this occurring have already been devised. Amongst them are COIN [34] and myopic heuristics [6]. However, these methods require knowledge of the reward function or the joint action. Potential-based reward shaping can similarly increase the probability of convergence to the optimal Nash Equilibrium provided a good heuristic, but does so without requiring either of these specific pieces of knowledge which are commonly unavailable in MAS applications.

Both applications of potential-based reward shaping were published with no consideration of whether the proofs of guaranteed policy invariance hold in multi-agent reinforcement learning or how they affect the joint policy at time of convergence. Starting in the following section, our contribution fills this gap in knowledge and provides the theoretical results to explain these previous empirical studies.

4. THEORY

To discuss the implications of using potential-based reward shaping in MAS we must consider the differences between single-agent and multi-agent reinforcement learning. SGs, unlike MDPs, share amongst all agents a common transition function and common states but neither of these are

affected by shaping the reward function of one or more of the agents. Although the agents may change their own policy and alter their exploration path due to the additional potential-based reward, this does not change the dynamics (transition function or states) of the environment, nor the set of actions the agent can take.

In fact the only elements of a SG to change when one or more agent implements potential-based reward shaping are the individual reward functions of those agents. If, as we will later show to be true in Section 4.2, these alterations to the individual reward functions do not change the best response policy of a shaped agent given a fixed set of policies followed by all other agents, the Nash Equilibria of the underlying SG remain constant regardless of how many agents are using potential-based reward shaping.

Formally, this argument will be completed by showing, in the following sub-section, that potential-based reward shaping in MAS is equivalent to Q-table initialisation and then, in Section 4.2, that it does not alter the Nash Equilibria of the MAS. Both of these findings, as we will discuss in Section 4.3, has implications for the eventual policy that will be converged upon.

4.1 Potential-Based Reward Shaping And Q-Value Initialisation Are Equivalent

The proof of Wiewiora [33] of the equivalence of potential-based reward shaping and Q-value initialisation was published in the context of single agent problem domains but also holds for problem domains with multiple individual learners.

From [33] we quote:

Theorem 1 Given the same sequence of experiences during learning, $\Delta Q(s, a)$ always equals $\Delta Q'(s, a)$.

where $Q(s, a)$ is the modelled value function of an agent learning with potential-based reward shaping and $Q'(s, a)$ is the modelled value function of an agent learning with Q-value initialisation.

The original proof uses a fixed sequence of experiences for both agents. The theory can be extended to multiple individual learners simply by extending the definition of the sequence experienced from the 4-tuple $\langle s, a, r, s' \rangle$ to the $2n + 2$ -tuple $\langle s, a_1, a_2, \dots, a_n, r_1, r_2, \dots, r_n, s' \rangle$. Using the extended sequence and the inductive proof from [33] the following proves that Theorem 1 holds also for multi-agent reinforcement learning.

Proof By Induction

Consider any arbitrary agent i from the set of all agents. As before, $Q(s, a)$ is the modelled value function when the agent is learning with potential-based reward shaping and $Q'(s, a)$ is the modelled value function had the same agent learnt without reward shaping but with Q-value initialisation. The former agent will later be referred to as L and the latter as L' .

Agent L will update its Q-values by the rule:

$$Q_i(s, a) \leftarrow Q_i(s, a) + \underbrace{\alpha (r_i + F(s, s') + \gamma \max_{a'} Q_i(s', a') - Q_i(s, a))}_{\delta Q_i(s, a)} \quad (4)$$

where $F(s, s')$ is the potential-based reward shaping function and $\delta Q_i(s, a)$ is the amount (scaled by α) that the Q value will be updated by. The current Q-values of Agent L can be represented formally as the initial value plus the change since:

$$Q_i(s, a) = Q_i^0(s, a) + \Delta Q_i(s, a) \quad (5)$$

where $Q_i^0(s, a)$ is agent i 's initial Q-value of state-action pair (s, a) . Similarly agent L' updates its Q-values by the rule:

$$Q'_i(s, a) \leftarrow Q'_i(s, a) + \alpha \underbrace{(r_i + \gamma \max_{a'} Q'_i(s', a') - Q'_i(s, a))}_{\delta Q'_i(s, a)} \quad (6)$$

And its current Q-values can be represented formally as:

$$Q'_i(s, a) = Q_i^0(s, a) + \Phi(s) + \Delta Q'_i(s, a) \quad (7)$$

where $\Phi(s)$ is the potential for state s .

Base Case

Before either agent experiences anything, the Q-tables of L and L' are both their respective initial values, and therefore both ΔQ_i and $\Delta Q'_i$ are uniformly zero.

Inductive Case

Assuming $\Delta Q_i = \Delta Q'_i$, both L and L' will be updated by the same amount in response to experience $\langle s, a_1, a_2, \dots, a_n, r_1, r_2, \dots, r_n, s' \rangle$. First consider the update performed by L :

$$\begin{aligned} \delta Q_i(s, a) &= r_i + F(s, s') + \gamma \max_{a'} Q_i(s', a') - Q_i(s, a) \\ &= r_i + \gamma \Phi(s') - \Phi(s) \\ &\quad + \gamma \max_{a'} (Q_i^0(s', a') + \Delta Q_i(s', a')) \\ &\quad - Q_i^0(s, a) - \Delta Q_i(s, a) \end{aligned} \quad (8)$$

Now consider the update performed by L' :

$$\begin{aligned} \delta Q'_i(s, a) &= r_i + \gamma \max_{a'} Q'_i(s', a') - Q'_i(s, a) \\ &= r_i + \gamma \max_{a'} (Q_i^0(s', a') + \Phi(s') + \Delta Q'(s', a')) \\ &\quad - Q_i^0(s, a) - \Phi(s) - \Delta Q'(s, a) \\ &= r_i + \gamma \max_{a'} (Q_i^0(s', a') + \Phi(s') + \Delta Q(s', a')) \\ &\quad - Q_i^0(s, a) - \Phi(s) - \Delta Q(s, a) \\ &= r_i + \gamma \Phi(s') - \Phi(s) \\ &\quad + \gamma \max_{a'} (Q_i^0(s', a') + \Delta Q_i(s', a')) \\ &\quad - Q_i^0(s, a) - \Delta Q_i(s, a) \\ &= \delta Q_i(s, a) \end{aligned} \quad (9)$$

Therefore, the Q-tables of both L and L' are both updated by the same value and so ΔQ_i and $\Delta Q'_i$ remain equal. \square

Given that Theorem 1 of [33] holds for the multi-agent context then so too does Theorem 2, again quoted from [33]:

Theorem 2 If L and L' have learnt on the same sequence of experiences and use an advantage-based policy, they will have an identical probability distribution for their next action.

where an advantage-based policy is one that chooses actions based not on the absolute magnitude of the Q-values but on their relative differences within the current state. Examples of advantage-based policies include greedy, ϵ -greedy and Boltzmann soft-max.

This is immediately apparent when considering both $\Delta Q_i = \Delta Q'_i$ from Theorem 1 and Equations 5 and 7. As the difference between the Q-values of agent L and agent L' are the potential of the state, the difference is consistent across all actions in any given state. Therefore, the actions maintain the same relative differences allowing an advantage-based policy to make the same action decisions.

Effectively, at any time in learning L and L' will behave the same way (make the same decisions with the same probabilities). To conclude, whether an agent is shaped or initialised it will have the same effect on all other agents in the environment, the learning dynamics are not changed by using one method or the other and the agents as a collective whole will converge or not upon the same joint policy regardless of whether the agent was shaped or initialised.

Finally, although the proof here was written specifically for Q-learning, this was simply in keeping with the original work of [33]. In single-agent problem domains the equivalence of Q-table initialisation and potential-based reward shaping can be proven also in SARSA and other temporal difference algorithms [33]. Similar extensions to multi-agent, as above, are possible also for these extensions.

4.2 Potential-Based Reward Shaping Does Not Alter The Nash Equilibria Of A Stochastic Game

As already established the common goal of MARL is a Nash Equilibrium. The typical concern of modifying a reward function is that the original goals of the agent will be altered. Ng showed previously that in the single-agent context, the optimum policy was unchanged by the introduction of reward shaping provided the function was potential-based [20]. To extend this to MARL we must now consider whether implementing the same reward shaping in one or more agents in a SG will alter its points of equilibrium.

Formally a Nash Equilibrium in a SG is:

$$\forall i \in 1 \dots n, \pi_i \in \Pi_i | R_i(\pi_i^{NE} \cup \pi_{-i}^{NE}) \geq R_i(\pi_i \cup \pi_{-i}^{NE}) \quad (10)$$

where n is the number of agents, Π_i is the set of all possible policies of agent i , R_i is the reward function for agent i , π_i^{NE} is a specific policy of agent i and π_{-i}^{NE} is the joint policy of all agents except agent i following their own fixed specific policy. If the inequality holds for all agents, the joint policy of each agent following its policy π_i^{NE} is a Nash Equilibrium.

Now consider any arbitrary agent i from the set of all agents. For the inequality above to hold for agent i , we must consider the set Π_i^{NE} of all joint policies consisting of each possible policy of agent i combined with π_{-i}^{NE} . Formally, this set contains:

$$\forall \pi_i \in \Pi_i | (\pi_i \cup \pi_{-i}^{NE}) \quad (11)$$

Each fixed joint policy in the set Π_i^{NE} will generate a fixed infinite sequence of experiences when followed consistently from the current state s_0 of the form:

$$\begin{aligned} \bar{s} = & s_0, a_{0,0}, a_{0,1}, \dots, a_{0,n}, r_{0,0}, r_{0,1}, \dots, r_{0,n}, \dots, \\ & s_\infty, a_{\infty,0}, a_{\infty,1}, \dots, a_{\infty,n}, r_{\infty,0}, r_{\infty,1}, \dots, r_{\infty,n}, \dots \end{aligned} \quad (12)$$

where s_j is the state at time j , $a_{j,i}$ is the action taken by agent i at time j and $r_{j,i}$ is the reward received by agent i at time j .

Then using the proof of [1], we can show the difference of the return received by agent i when following any arbitrary fixed sequence with or without potential-based reward shaping is the potential of the state s_0 .

Proof

The return for agent i when experiencing sequence \bar{s} in a discounted framework without shaping is:

$$U_i(\bar{s}) = \sum_{j=0}^{\infty} \gamma^j r_{j,i} \quad (13)$$

Now consider the same agent but with a reward function modified by adding a potential-based reward function. The return of the shaped agent experiencing the same sequence \bar{s} is:

$$\begin{aligned} U_{i,\Phi}(\bar{s}) &= \sum_{j=0}^{\infty} \gamma^j (r_{j,i} + F(s_j, s_{j+1})) \\ &= \sum_{j=0}^{\infty} \gamma^j (r_{j,i} + \gamma\Phi(s_{j+1}) - \Phi(s_j)) \\ &= \sum_{j=0}^{\infty} \gamma^j r_{j,i} + \sum_{j=0}^{\infty} \gamma^{j+1} \Phi(s_{j+1}) - \sum_{j=0}^{\infty} \gamma^j \Phi(s_j) \\ &= U_i(\bar{s}) + \sum_{j=1}^{\infty} \gamma^j \Phi(s_j) - \Phi(s_0) - \sum_{j=1}^{\infty} \gamma^j \Phi(s_j) \\ &= U_i(\bar{s}) - \Phi(s_0) \end{aligned} \quad (14)$$

□

Therefore, any policy that previously maintained the inequality of Equation 10 will still maintain the inequality. Formally, and more strictly we can conclude:

$$\begin{aligned} \forall \pi_i \in \Pi_i \mid & (R_i(\pi_i^{NE} \cup \pi_{-i}^{NE}) \geq R_i(\pi_i \cup \pi_{-i}^{NE})) \leftrightarrow \\ & (R_{i,\Phi}(\pi_i^{NE} \cup \pi_{-i}^{NE}) \geq R_{i,\Phi}(\pi_i \cup \pi_{-i}^{NE})) \end{aligned} \quad (15)$$

where $R_{i,\Phi}$ is the reward function of agent i when receiving both the environmental reward and the potential-based reward shaping.

As implementing reward shaping only affects the reward function of that agent, the remaining agents will also still maintain the same policies as part of the Nash Equilibria. Whether the group will converge to this point depends on the learning algorithm used and is outside of this proof. However, it suffices to say that regardless of how many agents in the MAS are or are not implementing potential-based reward shaping the points of equilibrium will remain constant.

4.3 Potential-Based Reward Shaping Alters Exploration

In Section 4.1 we showed that an agent in a MAS receiving potential-based reward shaping is equivalent to one whose Q-table was initialised with each state s set to the potential $\Phi(s)$ of that state. However, the implications of this proof in a MAS extend past showing that two methods of introducing domain knowledge are equivalent. Instead, it is worth considering the results of [32], in which Wellman and Hu showed that the joint policy converged upon in a learning MAS was highly sensitive to initial belief. This clearly applies directly to Q-table initialisation, where the initial values directly represent some initial belief, and therefore, given that we have shown the equivalence between initialisation and shaping, also applies to potential-based reward shaping. This can be reasoned intuitively by considering the following.

The MDP of an agent deployed in a common environment with other learning agents does not hold the Markov property as the transition probabilities are subject to change with the unseen but changing policies of the other agents. Therefore, the convergence to optimal policy guarantees of Q-learning do not hold. This has been demonstrated empirically in multi-agent reinforcement applications with multiple Q-learners converging to sub-optimal joint policies [2].

Shaping alters the path of exploration an agent takes. In single-agent reinforcement learning, as convergence to the optimal policy is guaranteed, this only affects the time taken to reach convergence. If a good heuristic, is used the time will be reduced as the number of sub-optimal actions taken will be reduced, but similarly if a bad heuristic is used the agent will take longer to converge to the optimal policy.

The concept of an optimal policy in MAS is not as clear. We have identified Nash Equilibrium as the typical goal of multi-agent reinforcement learning, but this does not necessarily identify a single goal. Most applications, with the exception of the very trivial, will have multiple points of equilibrium. Multiple individual learners will converge to one of these equilibrium, but whether it will be the optimum cannot be guaranteed [6]

With multiple agents in the same environment, altering the exploration of one will change the experiences of all agents [12, 13]. The change in actions chosen by even just one agent now receiving potential-based reward shaping will result in different state transitions. The agents will then explore different areas of the joint policy space and, with multiple points of equilibrium possible, may converge to a different equilibrium then had the agent not received the reward shaping and subsequently not have altered its individual exploration path.

Therefore, in multi-agent problem domains, without the guarantee of convergence to a single optimum goal, shaping can lead to convergence on a different joint policy. This was empirically demonstrated by Babes and Littman [2], where a shaped agent was able to lead a non-shaped agent to convergence on a joint policy of higher average reward. When shaping one or more agents in an environment with multiple learning agents, a good heuristic will encourage higher global utility similar to how in single-agent problem domains the use was preferably to reduce the time taken to converge. Unfortunately, the techniques can also have a detrimental effect encouraging miscoordination and/or lead the agents to converge on a less beneficial joint policy by directing the

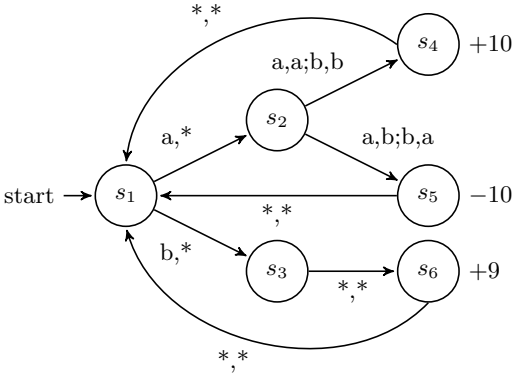


Figure 1: Boutilier's Coordination Game

agents away from frequently, or possibly ever, experiencing the equilibrium reached by non-shaped agents and instead trapping them in a sub-optimal point of equilibrium.

To support and illustrate these claims the following section will present an empirical study that is typically characteristic of implementing potential-based reward shaping in a MAS.

5. EMPIRICAL DEMONSTRATION

To demonstrate the theorised effects of potential-based reward shaping, an empirical study of a game based on Boutilier's coordination game [4] will be presented here.

The game, illustrated in Figure 1, has six stages and two agents, each capable of two actions (a or b). The first agent's first action choice in each episode decides if the agents will move to a state guaranteed to reward them minimally (s_3) or to a state where they must co-ordinate to receive the highest reward (s_2). However, in state s_2 the agents are at risk of receiving a large negative reward if they do not choose the same action.

In Figure 1, each transition is labelled with one or more action pairs such that the pair $a,*$ means this transition occurs if agent 1 chooses action a and agent 2 chooses either action. When multiple action pairs result in the same transition the pairs are separated by a semicolon($;$).

The game has three joint policy Nash Equilibria; the joint policy of opting for the safety state s_3 or the two joint policies of moving to state s_2 and coordinating on both choosing a or b . Any joint policy receiving the negative reward is not a Nash Equilibrium, as the first agent can choose to change its first action choice and so receive a higher reward by instead reaching state s_3 .

Three sets of agents will be the focus of these experiments. All agents, in all sets, will learn by Q-learning with an ϵ -greedy policy and discount factor (γ) of 1. One set will receive no reward shaping, to illustrate the average performance without heuristic knowledge, another set will receive potential-based reward shaping from a good heuristic whilst the final set receives shaping from a poor heuristic.

The good heuristic, designed to encourage co-operation, gives states s_1 , s_2 and s_4 the potentials 5, 10 and 15 respectively. All other states receive a potential of 0. Therefore, any transition from states s_1 to s_2 or s_2 to s_4 will receive an additional reward of +5 but transitioning instead from state

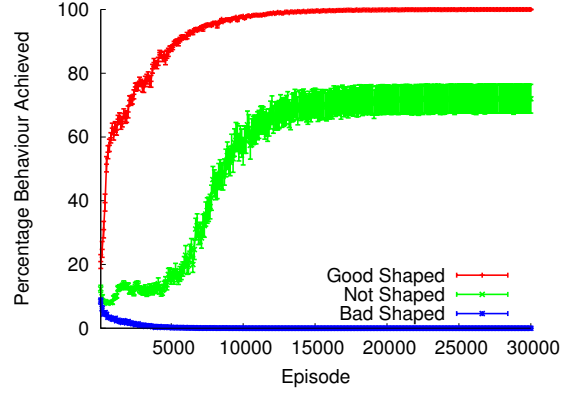


Figure 2: Optimal Nash Equilibrium

s_2 to s_5 will receive an additional reward of -10 . Alternatively, the bad heuristic is designed to encourage miscoordination and so potentials of 5, 10 and 15 are given instead to states s_1 , s_2 and s_5 respectively. Again all other states receive a potential of 0.

Our experimental results are intended to show, provided a good heuristic, the increased probability of converging to the joint policies of higher global utility (those achieving coordination in state s_2). Alternatively, provided a bad heuristic, the agents will demonstrate that the Nash Equilibria have not changed and so converge still to one of the three original joint policy Nash Equilibria.

5.1 Results

All experiments were run for 100,000 episodes (300,000 action choices) and repeated 100 times. The results, illustrated in Figures 2, 3 and 4, plot the mean percentage of the last 100 episodes performing the optimal, safety and sub-optimal joint policies respectively. All figures include error bars illustrating the standard error from the mean. For clarity, graphs are plotted only up to 30,000 episodes as by this time all experiments had converged to a stable joint policy.

Figure 2 shows that, for this relatively simple game, multiple individual learners alone can only converge to the optimal behaviour 72% of the time. Whereas, provided a good heuristic, potential-based reward shaping can increase the probability of convergence to this Nash Equilibrium to 100%.

As theorised, provided a bad heuristic, the effect on the global utility can be detrimental. The probability of achieving optimal behaviour, with a potential function encouraging miscoordination, rapidly drops and converges on 0%.

Instead, as illustrated by Figure 3, the poorly shaped agents converge to the safety Nash Equilibrium. Despite the miscoordination state (s_5) receiving the largest potential, the agents do not converge to the sub-optimal behaviour, as illustrated by Figure 4.

Figure 4, highlights that agents with no shaping or potential-based reward shaping never converge to consistently perform the sub-optimal joint policy. This is because miscoordination in this game is not a Nash Equilibrium, both with and without potential-based reward shaping. Regardless of which joint policy is encouraged, if the additional reward is potential based, the Nash Equilibria remain constant.

However, Figure 4 illustrates the behaviour of an addi-

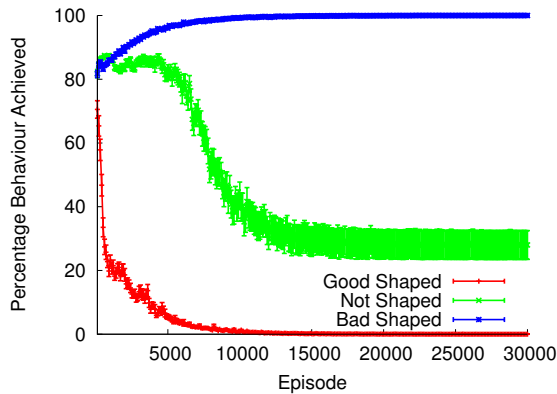


Figure 3: Safety Nash Equilibrium

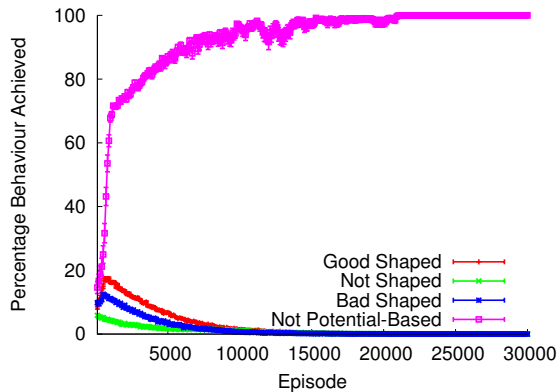


Figure 4: Sub-Optimal Behaviour

tional set of agents. These agents receive an additional reward on state transitions that is not potential based. Specifically they are rewarded 5, 10 and 30 upon entering states s_1 , s_2 and s_5 respectively. These agents converge to a joint policy representative of the sub-optimal behaviour. This has occurred because if additional rewards are not potential-based they can change the Nash Equilibria of a SG.

Finally, the learning performance of both sets of shaped agents favourably supports the use of potential-based reward shaping in MAS. The poorly shaped agents converge to the safety Nash Equilibrium after just 10,000 episodes whilst without shaping it takes agents 29,000 episodes to converge. More significantly, after only 2000 episodes agents receiving reward shaping from a good heuristic are more likely to achieve the optimal Nash Equilibrium than non-shaped agents ever will.

6. CONCLUSION

In conclusion, this paper shows how two fundamental papers in single-agent reward shaping [20, 33] can be extended to provide similar guarantees in multi-agent reinforcement learning.

Specifically, we have proven that a potential-based shaped agent is still equivalent to an agent with initial Q-values set to the potential of each state regardless of how many exist within the same environment.

Furthermore, we have also proven that rewarding any

number of agents within a MAS with additional potential-based rewards has no subsequent effect on the Nash Equilibria of the underlying SG.

Potential-based reward shaping affects the exploration of the shaped agent. Therefore, it can change the joint policy converged upon as even just one agent’s modified exploration can sufficiently redirect the search of joint policy space to converge to a different point of equilibrium.

Although the agents may now converge to a different joint policy, the latter of the two proofs guarantees that the new joint policy was also a goal of the unshaped agents.

Whether the goal achieved is the Nash Equilibrium of highest global utility, is dependent on the agents’ learning algorithms. With multiple individual learners, no guarantee of convergence to the highest utility Nash Equilibrium is provided. However, potential-based reward shaping can, dependent on the heuristic, either increase or decrease the probability of converging to equilibria of higher global utility as demonstrated in our empirical study.

Given a joint action learner guaranteed under fixed conditions to converge, such as NashQ [11], it is possible to construct similar proofs as those shown here. Agents learning by joint action and receiving potential-based reward shaping benefit from consistent Nash Equilibria, modified exploration to decrease the number of sub-optimal action decisions and guaranteed convergence.

It is also the authors’ expectation that potential-based advice [8], an extension of potential-based reward shaping to include heuristics based on actions as well as states, could similarly be extended to guarantee consistent Nash Equilibria when applied to multi-agent reinforcement learning. Recent empirical work supports these expectations [7].

The work here has been based entirely in fully observable problem domains, which some may consider uncharacteristic of MAS. However, by shaping agents based on the potential of observations (as opposed to fully observed states) the same arguments and proofs can be used to show similar theoretical expectations in partially observable problem domains. Namely, the Nash Equilibria of a partially observable problem domain would remain the same but the agents exploration will alter and so convergence may be to a different point of equilibrium or, given an unsuitable heuristic, may not converge at all.

In closing, adding potential-based reward shaping to multiple individual learners does not alter the Nash Equilibria but can, provided suitable heuristics, increase the probability of convergence to a higher global utility and decrease the time to convergence.

7. ACKNOWLEDGEMENTS

We would like to thank M.Babes, M.Littman and M.Grzes for their input during the development of these ideas and C.Poskitt for his time spent proof reading.

8. REFERENCES

- [1] J. Asmuth, M. Littman, and R. Zinkov. Potential-based shaping in model-based reinforcement learning. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence*, pages 604–609, 2008.
- [2] M. Babes, E. de Cote, and M. Littman. Social reward shaping in the prisoner’s dilemma. In *Proceedings of*

- the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, volume 3, pages 1389–1392, 2008.
- [3] D. P. Bertsekas. *Dynamic Programming and Optimal Control (2 Vol Set)*. Athena Scientific, 3rd edition, 2007.
 - [4] C. Boutilier. Sequential optimality and coordination in multiagent systems. In *International Joint Conference on Artificial Intelligence*, volume 16, pages 478–485. Citeseer, 1999.
 - [5] L. Busoniu, R. Babuska, and B. De Schutter. A Comprehensive Survey of MultiAgent Reinforcement Learning. *IEEE Transactions on Systems Man & Cybernetics Part C Applications and Reviews*, 38(2):156, 2008.
 - [6] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 746–752, 1998.
 - [7] S. Devlin, M. Grześ, and D. Kudenko. Multi-agent, potential-based reward shaping for RoboCup KeepAway. In *Proceedings of The Tenth Annual International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2011.
 - [8] G. C. Eric Wiewiora and C. Elkan. Principled methods for advising reinforcement learning agents. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.
 - [9] J. Filar and K. Vrieze. *Competitive Markov decision processes*. Springer Verlag, 1997.
 - [10] M. Grześ and D. Kudenko. Plan-based reward shaping for reinforcement learning. In *Proceedings of the 4th IEEE International Conference on Intelligent Systems (IS'08)*, pages 22–29. IEEE, 2008.
 - [11] J. Hu and M. Wellman. Nash Q-learning for general-sum stochastic games. *The Journal of Machine Learning Research*, 4:1039–1069, 2003.
 - [12] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in cooperative multi-agent systems. In *Proceedings of the National Conference on Artificial Intelligence*, pages 326–331. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.
 - [13] S. Kapetanakis and D. Kudenko. Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. pages 119–131, 2004.
 - [14] M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the eleventh international conference on machine learning*, volume 157, page 163. Citeseer, 1994.
 - [15] M. Littman. Friend-or-foe Q-learning in general-sum games. In *Machine Learning - International Workshop then Conference*, pages 322–328, 2001.
 - [16] B. Marthi. Automatic shaping and decomposition of reward functions. In *Proceedings of the 24th International Conference on Machine learning*, page 608. ACM, 2007.
 - [17] M. Mataric. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.
 - [18] M. Mihaylov, K. Tuyls, and A. Nowé. Decentralized Learning in Wireless Sensor Networks. *Adaptive and Learning Agents*, pages 60–73, 2009.
 - [19] J. Nash. Non-cooperative games. *Annals of mathematics*, 54(2):286–295, 1951.
 - [20] A. Y. Ng, D. Harada, and S. J. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, pages 278–287, 1999.
 - [21] J. Peters, S. Vijayakumar, and S. Schaal. Reinforcement learning for humanoid robotics. In *Proceedings of Humanoids2003, Third IEEE-RAS International Conference on Humanoid Robots*, 2003.
 - [22] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994.
 - [23] J. Randlev and P. Alstrom. Learning to drive a bicycle using reinforcement learning and shaping. In *Proceedings of the 15th International Conference on Machine Learning*, pages 463–471, 1998.
 - [24] Y. Shoham, R. Powers, and T. Grenager. If multi-agent learning is the answer, what is the question? *Artificial Intelligence*, 171(7):365–377, 2007.
 - [25] P. Stone and M. Veloso. Team-partitioned, opaque-transition reinforcement learning. In *Proceedings of the third annual conference on Autonomous Agents*, pages 206–212. ACM, 1999.
 - [26] R. Sutton. Generalization in Reinforcement Learning: Successful Examples Using Sparse Coarse Coding. *Advances in Neural Information Processing Systems*, pages 1038–1044, 1996.
 - [27] R. S. Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, 1984.
 - [28] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
 - [29] M. Tan. Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Proceedings of the Tenth International Conference on Machine Learning*, volume 337, 1993.
 - [30] K. Tumer and N. Khani. Learning from actions not taken in multiagent systems. *Advances in Complex Systems (ACS)*, 12(04):455–473, 2009.
 - [31] X. Wang and T. Sandholm. Reinforcement learning to play an optimal Nash equilibrium in team Markov games. *Advances in neural information processing systems*, pages 1603–1610, 2003.
 - [32] M. Wellman and J. Hu. Conjectural equilibrium in multiagent learning. *Machine Learning*, 33(2):179–200, 1998.
 - [33] E. Wiewiora. Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19(1):205–208, 2003.
 - [34] D. Wolpert and K. Tumer. An introduction to collective intelligence. Technical Report cs.LG/9908014, NASA Ames Research Center, 1999.
 - [35] M. Wooldridge. *An Introduction to MultiAgent Systems*. John Wiley and Sons, 2002.

Evolving Subjective Utilities: Prisoner's Dilemma Game Examples

Koichi Moriyama

Satoshi Kurihara

Masayuki Numao

The Institute of Scientific and Industrial Research, Osaka University
8-1, Mihogaoka, Ibaraki, Osaka, 567-0047, Japan
{koichi, kurihara, numao}@ai.sanken.osaka-u.ac.jp

ABSTRACT

We have proposed the *utility-based Q-learning* concept that supposes an agent internally has an emotional mechanism that derives subjective utilities from objective rewards and the agent uses the utilities as rewards of Q-learning. We have also proposed such an emotional mechanism that facilitates cooperative actions in Prisoner's Dilemma (PD) games.

However, this mechanism has been designed and implemented manually in order to *force* the agents to take cooperative actions in PD games. Since it seems slightly unnatural, this work considers whether such an emotional mechanism exists and where it comes from. We try to *evolve* such mechanisms that facilitate cooperative actions in PD games by conducting simulation experiments with a genetic algorithm, and we investigate the evolved mechanisms from various points of view.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*intelligent agents, multiagent systems*

General Terms

Experimentation

Keywords

Reward structures for learning; Multiagent Learning; Evolution, adaptation; Game theory

1. INTRODUCTION

In this paper, we consider a learning agent that chooses actions seeming appropriate. The most popular learning method for agents is reinforcement learning [14]. In reinforcement learning, an agent is given rewards from the environment according to its actions and the states of the environment, and the agent learns to take actions that maximize the rewards.

If there is only one agent in the world, it can take actions that maximize its own rewards. However, in a multiagent environment consisting of multiple agents, the maximization becomes impossible because of interactions among agents.

Cite as: Evolving Subjective Utilities: Prisoner's Dilemma Game Examples, Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 233-240.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

In this case, cooperative behaviors like yielding are necessary, but reinforcement learning as a reward-maximizer fails to learn such behaviors.

When we consider the most intelligent agents, i.e., humans, they take cooperative behaviors in such cases. For example, Fehr and Schmidt [3] tried to explain human behaviors by introducing “inequity-aversion” assumption into their models. This assumption represented a feeling of aversion to the difference from others and prevented the models from outsmarting others. Rilling et al. [11] reported that, when humans took cooperative behaviors, reward processing areas in their brains were positively activated. They inferred that the cooperative behaviors themselves became a kind of rewards in their brains.

Based on them, we have proposed the *utility-based Q-learning* concept [8]. This concept supposes that the agent internally has an *emotional mechanism* that derives *subjective utilities* from *objective rewards*, and it uses the utilities as rewards of Q-learning [16], a representative method of reinforcement learning. We have also proposed such an emotional mechanism that facilitates cooperative actions in Prisoner's Dilemma (PD) games [8]. PD games [1, 9] are the most famous two-person two-action game in game theory, in which if the two players try to maximize individual payoffs without cooperating with each other, they obtain less payoffs than those they would obtain when cooperating with each other. The mechanism we have proposed derives a larger subjective utility than the payoff each agent is given when the agents take mutual cooperation in PD games. The agents learning by the utilities take cooperation afterward.

However, this utility deriving function has been designed and implemented manually in order to *force* the agents to take cooperative actions in PD games. Since it seems slightly unnatural, this work considers whether such an emotional mechanism exists and where it comes from. We try to *evolve* such mechanisms that facilitate cooperative actions in PD games by conducting simulation experiments with a genetic algorithm. Notice that the objective of this work is not to acquire a strategy itself in PD games, but to know whether such an emotional mechanism evolves.

This paper consists of six sections. Section 2 introduces PD games, Q-learning, the utility-based Q-learning concept, and genetic algorithms, which are used in the following sections. In Section 3, we see the experiment scheme to obtain the emotional mechanisms for PD games by evolution. In Section 4, we investigate the emotional mechanisms we obtained from the experiment. After we check several related works in Section 5, this paper is summarized in Section 6.

Table 1: Prisoner’s Dilemma game

$A \setminus B$	C	D
C	R, R	S, T
D	T, S	P, P

2. BACKGROUNDS

This section introduces PD games, Q-learning, the utility-based Q-learning concept, and genetic algorithms, which are used in the following sections. We use the terms “payoff” and “reward” in the same sense in this paper.

2.1 Prisoner’s Dilemma Games

A Prisoner’s Dilemma (PD) game [1, 9] is a two-person two-action game and is often shown by a bimatrix, called a payoff matrix (Table 1). Each player has two actions C (cooperation) and D (defection). The players A and B choose actions from rows and columns, respectively. After choosing actions, each player obtains a payoff in Table 1. For example, when A and B choose C and D , respectively, A and B obtain payoffs S and T , respectively. Hereafter, (X, Y) stands for the pair of actions of both players such that X and Y are the actions of A and B , respectively.

PD has the following relations among payoffs:

$$T > R > P > S \quad \text{and} \quad 2R > T + S.$$

Under these relations, each player obtains a larger payoff when he/she chooses D regardless of the opponent’s action. As a result, the action pair becomes (D, D) and each player obtains a payoff P . However, it is more desirable for both players to choose the first actions (C, C) and obtain R which is larger than P . Since rational decisions of both players give a worse result, this game is called “dilemma”.

2.2 Q-learning

Suppose an agent senses a state $s_t \in \mathcal{S}$ and chooses an action $a_t \in \mathcal{A}(s_t)$ at a discrete time t . \mathcal{S} is a set of possible states in the environment and $\mathcal{A}(s_t)$ is a set of possible actions in the state s_t . After choosing an action, the agent receives a reward $r_{t+1} \in \mathbb{R}$ and senses a new state s_{t+1} . Q-learning [16] updates an *action value function* Q by the following rule to make it approach the true *value* under the optimal policy π^* , which is the expected sum of rewards discounted by $\gamma \in [0, 1)$ under π^* , i.e., $E_{\pi^*}(\sum_{k=0}^{\infty} \gamma^k r_{t+1+k})$.

$$Q_{t+1}(s, a) = \begin{cases} Q_t(s_t, a_t) + \alpha \delta_t & \text{if } (s, a) = (s_t, a_t), \\ Q_t(s, a) & \text{otherwise.} \end{cases}$$

$$\delta_t \equiv r_{t+1} + \gamma \max_{a' \in \mathcal{A}(s_{t+1})} Q_t(s_{t+1}, a') - Q_t(s_t, a_t).$$

$\alpha \in (0, 1]$ is a parameter called the learning rate and δ_t is called TD error that approaches 0 when $Q(s, a)$ approaches the true value of the action a in the state s under π^* . For all s and a , $Q(s, a)$ is proved to converge to the true value $Q^*(s, a)$ under π^* with probability one when (i) the environment has the Markov property, (ii) the agent visits all states and takes all actions infinitely, and (iii) α is decreased properly [16].

If the true value function Q^* is known, the agent can choose an optimal action a^* in a state s from Q^* by

$$a^* = \arg \max_{a'' \in \mathcal{A}(s)} Q^*(s, a'').$$

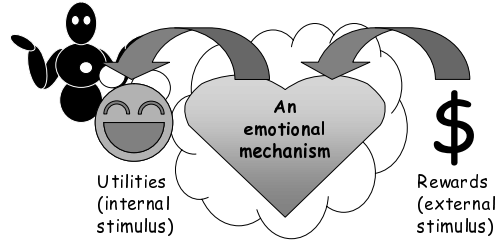


Figure 1: An emotional mechanism

However, if the agent always chooses such actions during learning, Q may converge to a local optimum because the agent may not visit all states. To avoid it, the agent usually uses a stochastic method like ϵ -greedy [14] to choose actions. ϵ -greedy method chooses either an action having the maximum Q with probability $1 - \epsilon$ or a random action with probability ϵ .

2.3 Utility-based Q-learning

The utility-based Q-learning concept [8] supposes that the agent internally has an emotional mechanism that derives *subjective utilities* from *objective rewards* (Figure 1), and it uses the utilities as rewards of Q-learning. Especially, in one-state Q-learning, we have also proposed such an emotional mechanism, namely a utility deriving function, that facilitates mutual cooperation in PD games [8]. This utility deriving function derives a utility u from a reward r ,

$$u \equiv \begin{cases} r + r' (\equiv R + r') & \text{if the actions are } (C, C), \\ r & \text{otherwise,} \end{cases} \quad (1)$$

such that

$$r' \geq \frac{P - (\alpha R + (1 - \alpha)S)}{\alpha} \quad (2)$$

and $\alpha \in (0, 1)$ is constant.

The utility u derived from Equation 1 makes the action value of cooperation larger than (or equal to) that of defection, i.e., $Q(C) \geq Q(D)$, after a single mutual cooperation. It comes theoretically from the number of consecutive mutual cooperation that is necessary to make $Q(C) \geq Q(D)$, when $Q(D)$ is the limit value after infinite mutual defections. See the paper [8] for details.

2.4 Genetic Algorithm

A genetic algorithm (GA) [5, 7] is an algorithm that simulates the evolution of creatures. GA uses many entities called *genomes* consisting of multiple *genes*. Each genome represents a solution of the target problem. GA finds a genome that maximizes a *fitness function*, which indicates goodness of a genome for the target problem, by iterating *selection*, *crossover*, and *mutation*. GA can be applied to many problems of which we can define fitness functions.

Simple GA first defines genomes and a fitness function from solution candidates and the objective of the problem, respectively. After that, GA executes the following procedure iteratively. One iteration is called one *generation*. Finally, a genome with the maximum fitness shows the best solution satisfying the objective of the problem.

1. Generate N genomes in a set named “current” randomly and prepare an empty set named “new”.

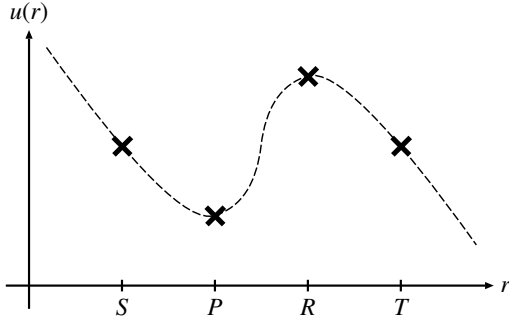


Figure 2: Assumption of a utility deriving function

2. Calculate the fitness of each genome in the current set.
3. According to the fitness, select two genomes from the current set (selection), copy them, and apply the following procedure to the copied genomes to create new genomes. After that, the new genomes are added to the new set.
 - Exchange several genes between the two genomes with probability p_c (crossover).
 - Modify each gene in each genome with probability p_m (mutation).
4. Until the size of the new set becomes N , repeat 3.
5. If the termination condition is not satisfied, clear out the current set and move all the contents of the new set to the current set. Back to 2.

3. EVOLVING UTILITIES

Although *the* utility-based Q-learning with the utility deriving function (Equation 1) was shown to bring out mutual cooperation [8], this function has been designed and implemented manually in order to *force* the agents to take cooperative actions in PD games. Since it seems slightly unnatural, this work considers whether such a utility deriving function, i.e., an emotional mechanism, exists and where it comes from. Humans, who may show mutual cooperation, appeared as the result of evolution. Therefore, we try to obtain such an emotional mechanism by evolutionary calculation using GA.

This work especially investigates how utility deriving functions evolve by GA when N agents having these functions play PD games in a round-robin fashion.

3.1 Genome Definition

In order to apply GA to evolving utility deriving functions, first we have to define genomes. Suppose $u(r)$ shows a utility deriving function that derives a utility u when the reward is r . If the payoffs $T > R > P > S$ in Table 1 become $u(R) > u(T)$, $u(S) > u(P)$, and $u(R) > u(P)$, it is obvious that the action C becomes superior to D . Since this function $u(r)$ can be depicted as like Figure 2, we assume that $u(r)$ is a cubic function, i.e.,

$$u \equiv u(r) \equiv ar^3 + br^2 + cr + d, \quad (3)$$

and the coefficients a, b, c, d are evolved by real-valued GA. Note that we do not assume that the cubic function itself

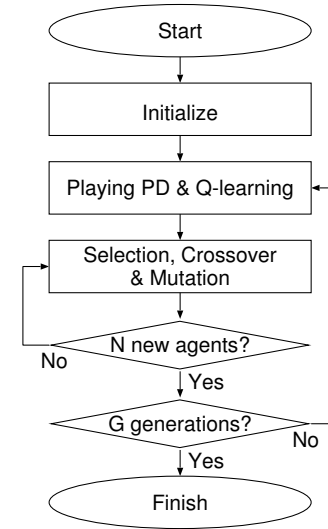


Figure 3: Flowchart of a run of the experiment

promotes cooperation. We simply choose it because it may be the simplest function that can represent the assumption of Figure 2¹.

3.2 Experiment Scheme

Let N and G be the number of agents in a generation and the total number of generations, respectively. Figure 3 shows the overview of a run of the experiment. This is based on the simple GA shown in Section 2.4.

Initialize: Two empty sets named “current” and “new” are prepared. N Q-learning agents each of which has its own u represented by a genome (Equation 3) are constructed and put into the current set. Each gene, i.e., the coefficients a, b, c, d of Equation 3, is a random real value in a finite interval. The action value functions Q of the agents are initialized.

Playing PD & Q-learning: N agents play iterative PD games in a round-robin fashion. Each play consists of M games. After each game, each agent executes utility-based Q-learning with the utility u derived from Equation 3.

Let r_{ij}^g be the sum of payoffs an agent i obtains in a play with an agent j at the generation g . After playing with all other agents, the agent i calculates the (raw) fitness $f_i^g = \sum_j r_{ij}^g$. Notice that the fitness is the sum of payoffs, not the sum of utilities.

Selection, Crossover & Mutation: Two agents are chosen from the current set by roulette wheel selection according to the *scaled* fitness that is calculated from f_i^g with a linear scaling method [5] with the coefficient ξ . That avoids premature convergence to extraordinary individuals at the beginning and emphasizes the differences among genomes at the end.

¹Naturally, there are other functions that can represent the assumption. We need to investigate the cases of using other functions.

The genomes of the selected agents are copied. The copied genomes are crossed, with probability p_c , by the uniform crossover that swaps genes of the genomes with probability 0.5. Then, each of the two genomes is mutated by adding a real value from a Gaussian distribution $N(\mu, \sigma)$ to each gene with probability p_m .

After that, two agents having the two new genomes are constructed and added to the new set. The action value functions Q of the new agents are initialized.

N new agents? Until the size of the new set becomes N , “Selection, Crossover & Mutation” is repeated.

G generations? Until evolving G generations, the current set is cleared out and all agents in the new set are moved to the current set. The whole process except “initialize” is repeated.

We use the following parameters in the experiment: the number of agents $N = 100$, the number of generations $G = 10000$, the number of games in a play $M = 1000$, the coefficient of the linear scaling method $\xi = 1.2$, the probability of crossover $p_c = 0.9$, and that of mutation $p_m = 0.01$. The interval for each gene is $[-10, 10]$ and the Gaussian distribution for mutation is $N(0, 1)$. We do not employ the elite strategy that copies the best genome into the next generation automatically.

The payoffs of PD game are identical with those used in the Axelrod’s tournament [1], i.e., $(T, R, P, S) = (5, 3, 1, 0)$.

The number of state of Q-learning is set to one. This means each agent does not remember the action sequences at all and thus each game becomes identical with a one-shot PD game for the agents. The parameters of Q-learning are identical with those in the previous paper [8], i.e., the learning rate $\alpha = 0.25$, the discount factor $\gamma = 0.5$, ε in ε -greedy method is 0.05, and the initial values of the action value function Q are all zero. Since $M \times (\varepsilon/2)^2 = 0.625$, both agents take an action that has less Q simultaneously around 0.625 times in a play.

GAlib² 2.4.7 is used as the implementation of real-valued GA.

4. RESULTS

The experiment described in Section 3.2 was conducted 100 runs. In this section, we investigate the results of the experiment.

4.1 Did mutual cooperation occur?

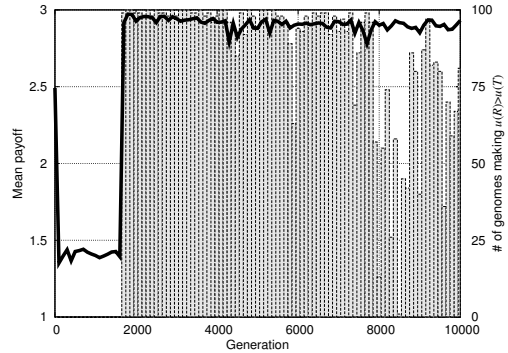
At the end of each run, the average payoff each agent obtained per game became more than 2.7 in 83 out of 100 runs. This means that mutual cooperation occurred because $(T + S)/2 = 2.5$. However, in the remaining 17 runs each agent obtained around 1.4 in the mean, which showed mutual defection was continuing.

Figure 4 shows the mean payoffs per game at each generation in a certain run. The function u of the best agent that obtained the highest payoff at the end of this run became

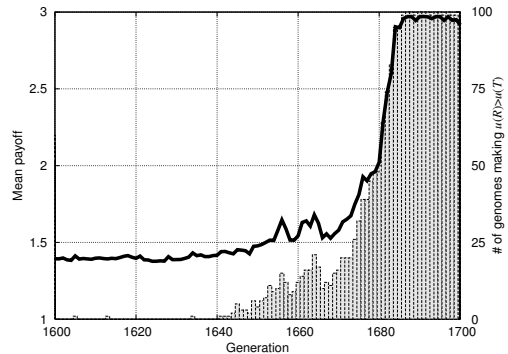
$$u(r) = -1.18073r^3 + 7.81789r^2 - 4.44985r - 10$$

and the agent obtained 2.95169 per game. There were eight agents that had same u , and they obtained more than 2.9 in

²<http://lancet.mit.edu/ga/>



(a) Overall view (plotting every 100 generations)



(b) Closeup at the phase transition (plotting every generation)

Figure 4: Mean payoffs per game (line with the left scale) and the number of genomes making $u(R) > u(T)$ (bars with the right scale) in a certain run

the mean. This u is shown by a solid line in Figure 5. It goes against the assumption of Figure 2 because $u(R) \equiv u(3) = 15.13175$ is slightly less than $u(T) \equiv u(5) = 15.60675$. It is also obvious that $u(S) \equiv u(0) < u(1) \equiv u(P)$. Thus, this u also gives the agent a PD game.

We can see in Figure 4(b) a kind of phase transition where the average payoffs rose suddenly in a short time of a few dozen games. Although it depended on runs when the phase transition occurred, the phase transition period was a few dozen games in most runs. This is discussed in Section 4.3.

It is clear that the first coefficient a in Equation 3 should be negative so as to realize the assumption of Figure 2. Although the coefficients of most genomes at the end of the 83 successful runs were negative, those of 16 out of 17 failed runs were positive. In the remaining one run, the mean payoffs per game around the end were fluctuating. It might be the beginning of the phase transition.

4.2 What property did u have?

In addition to u of the best agent at the end of a certain run, Figure 5 also shows u of the best agent at the 1700th generation of this run, which was just after the phase transition, by a dotted line. We can see that, at the 1700th generation, $u(R)$ was obviously larger than $u(T)$ as the assumption of Figure 2, but at the end of the run, as we saw before, $u(R)$ became less than $u(T)$.

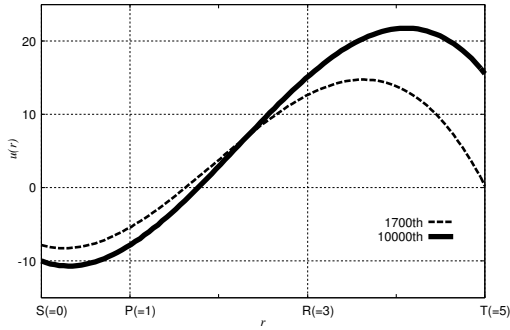


Figure 5: Utility deriving functions of agents obtaining the highest per-game payoff at the 1700th generation and at the 10000th generation in the run of Figure 4, respectively

Hence, let us investigate the relation between $u(R)$ and $u(T)$ in all generations. Figure 4 also shows the number of genomes that made $u(R) > u(T)$ by bars. Let us call such genomes “ R -preferring”. We can see that, most genomes were R -preferring until around the 8000th generation, but the number fluctuated after that. We have not found the reason why that fluctuation happens. We can only say about the run that, if the best genome was R -preferring at the start of decreasing, it was replaced by a non- R -preferring genome soon. On the other hand, interestingly, when the number was increasing, the best non- R -preferring genome continued to stay for a while and R -preferring genomes were prevailing in the middle level.

We investigated all 83 runs showing the phase transition and found the following two cases:

1. Most of all genomes were R -preferring at least once, like the example shown before (Figure 4), and
2. Only a small number of genomes were R -preferring, like an example shown in Figure 6.

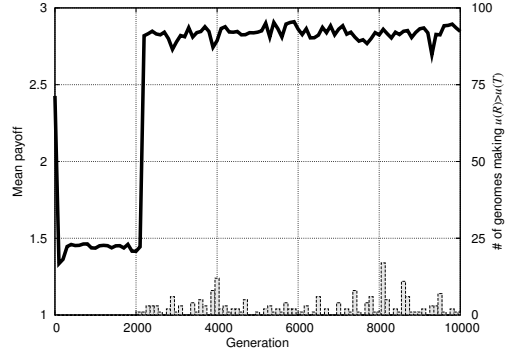
In order to eliminate the effect of random initial values, we ignore first 20 generations and analyze the remaining 9980 generations of the result of each run. It was 35 out of 83 runs that all genomes were R -preferring at least once and it was 19 out of 83 runs that R -preferring genomes were in the minority in all generations. It was only five out of 83 runs that the best genome were R -preferring at the end of the runs. Hence, in at least 30 runs, the best genome changed from an R -preferring one to a non- R -preferring one.

4.3 What happened at the phase transition?

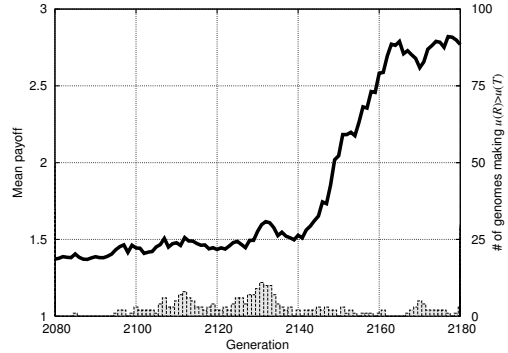
In Section 4.2, we investigated the number of R -preferring genomes in all generations. In this section, let us focus on the phase transition period and investigate the number of R -preferring genomes and generations in this period.

First the period is to be defined. Based on the results like Figures 4 and 6, we define the start and the end of the period as the generations when the average payoff first exceeded 1.6 and 2.7, respectively.

Under this definition, Figure 7 shows the length of the period. Y -axis shows the number of generations in log scale and x -axis shows each run sorted by the number of generations. From this figure, we can see that, in most runs, the



(a) Overall view (plotting every 100 generations)



(b) Closeup at the phase transition (plotting every generation)

Figure 6: A certain run in which only a small number of genomes made $u(R) > u(T)$

phase transition period was less than 100 generations, and the length seems to follow an exponential function.

Figure 8 shows the average numbers of R -preferring genomes during the period. In each run, the total number of R -preferring genomes during the period was divided by the number of generations of the period. X -axis shows each run sorted by the average number of genomes. We can see that this graph consists of two linear lines bending around the 37th, where the number of R -preferring genomes is around 10. Eighteen out of the aforementioned 19 runs of the “minority case”, in which R -preferring genomes were in the minority in all generations, were left side of the bending point. On the other hand, 31 out of the 35 runs of the “majority case”, in which all genomes were R -preferring at least once, were right side of the point. Therefore, the left side and the right side may correspond to the minority case and the majority case, respectively. It suggests that about a half of the phase transition is similar to the minority case. This is interesting because it disagrees with the assumption of Figure 2.

Figure 9 shows the relation between the number of R -preferring genomes in the phase transition period and the duration of the period. We can see the variance became smaller as the number of genomes increased.

More analyses are necessary to know the reason why mutual cooperation occurred even when R -preferring genomes were in the minority. Perhaps, since the condition $u(R) >$

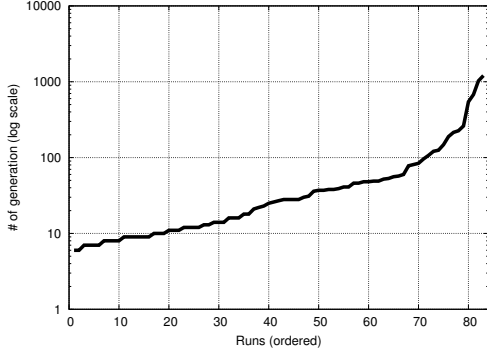


Figure 7: Number of generations needed for phase transition. x : each run (sorted by the number of generations), y : number of generations (log scale)

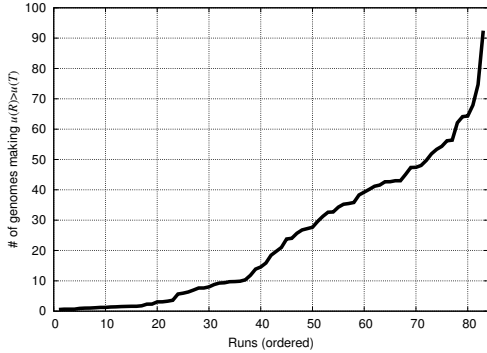


Figure 8: Average number of genomes that made $u(R) > u(T)$ in the phase transition period. x : each run (sorted by the average number), y : average number of genomes

$u(T)$ is not a necessary one but a sufficient one, genomes in such runs may have evolved so as to satisfy only (unknown) necessary conditions.

4.4 Relation between u and Formula 2

Formula 2 appeared in Section 2.3 is a kind of necessary condition to facilitate mutual cooperation in PD games. Hence, based on the discussion in Section 4.3, here we investigate the relation between the result of this work and Formula 2.

Assigning the learning rate $\alpha = 0.25$ of the experiment to Formula 2 gives the following formula:

$$r' \geq 4P - R - 3S.$$

From this formula, if $4P - R - 3S > 0$, i.e., $R < 4P - 3S$, r' should be larger than 0. This means that, in order to maintain mutual cooperation in one-state Q-learning, it is necessary to make the payoff larger. On the other hand, if $R \geq 4P - 3S$, the mutual cooperation can be maintained by the original payoff. Hence, we check whether the utilities u calculated by Equation 3 satisfied

$$u(R) \geq 4u(P) - 3u(S) \quad (4)$$

so as to know whether the mutual cooperation could be

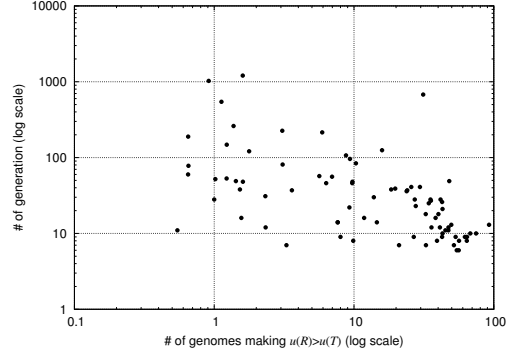


Figure 9: Relation between the number of genomes making $u(R) > u(T)$ in the phase transition period and the duration of the period. x : average number of genomes (log scale), y : number of generations (log scale)

maintained only by the evolved utilities.

The result shows that most of all genomes in most of all runs satisfied Formula 4 regardless of the phase transition. It is because, as $u(T)$ became much larger before the phase transition, $u(R)$ also became much larger than $u(P)$ and $u(S)$ ³. It may be a reason why mutual cooperation occurred even when R -preferring genomes were in the minority.

5. RELATED WORKS

There are several existing works in which agents learn strategies in PD games. Sandholm and Crites [12] conducted various experiments in which Q-learning agents played PD games and investigated whether or not mutual cooperation occurred. They reported that mutual cooperation did not occur when both agents did not take their past actions into account and that the parameters determining exploration rates had a major impact on the result when both agents used the past actions. Stimpson et al. [13] investigated what patterns the *satisficing strategy* produced in PD games. This strategy consisted of a parameter named *aspiration*, a simple behavior rule, and a simple update rule for aspiration. Under the behavior rule, the agent (i) continued to take the current action if it gave a larger payoff than aspiration, or (ii) took the other action otherwise. This work is interesting because it tried to derive a satisfactory result instead of an optimal result.

There are a lot of works that tried to obtain strategies of PD games by evolutionary algorithms. Here we see only a few of them. Axelrod [1] held two computer competitions of PD games, in which all of the attending programs played iterative PD games in a round-robin fashion. In both competitions, a strategy named Tit for Tat (TFT) became the best one according to the mean payoffs. After the competitions, he investigated what strategies of PD games evolved by GA [2]. First he specified that each locus of a gene indicated the sequence of past three actions and each gene indicated the next action. After that, he calculated some representative

³This may depend largely on the fact that u was a continuous function in which $u(T)$ and $u(R)$ were (somewhat) dependent. We also have to investigate the cases where u is a discrete function, in which they are completely independent.

strategies statistically from all of the attending programs of the second competition. He reported that, by playing PD games with these representative strategies, GA evolved strategies like TFT. Also, he reported that if all genomes in the genome set played PD games in a round-robin fashion, mutual cooperation occurred after growth and decay of defection. Fogel [4] also reported that cooperation occurred in PD games if genomes were defined as finite state automata. Vega-Redondo [15] introduced multiple sets of genomes. In each set, genomes played PD games with each other and evolved according to payoffs, and, simultaneously, each set was also under selection pressure according to the sum of payoffs of member genomes. Under this setting, he showed mutual cooperation was evolved even in one-shot PD games. Obviously, the selection of the sets themselves facilitated the cooperation.

Several works used both learning and evolution. Hingston and Kendall [6] introduced a kind of “switch” gene into the traditional genome which represented (fixed) next actions depending on the past action sequences. The switch gene determined whether the agent learned the opponent’s action model and changed its actions according to the model. They investigated how the genomes evolved by mutations and what strategy was spread. The result showed that the learning genomes did not increase so much. Quek et al. [10] proposed memetic learning and tried to obtain strategies of PD games. Memetic learning combined learning and evolution as an optimizer in an individual and a communicator between individuals, respectively. They reported that by combining learning and evolution, both compensated each other and, as a result, memetic learning could derive good strategies.

All of those works investigated strategies themselves for PD games. However, the work of this paper is not to derive strategies themselves for PD games, but to investigate whether an emotional mechanism that derives subjective utilities can evolve according to objective payoffs. This is a major difference between this work and those works.

6. CONCLUSION

The utility-based Q-learning concept supposes an agent internally has an emotional mechanism deriving subjective utilities from objective payoffs and it uses the utilities as rewards of Q-learning. In this work, we tried to obtain such an emotional mechanism by evolutionary computation using genetic algorithm (GA). Especially, we tried to evolve such mechanisms that facilitated cooperative actions in a game named Prisoner’s Dilemma (PD).

First we assumed that the mechanism could be represented by a cubic function and used real-valued GA to evolve its coefficients. We succeeded in obtaining mutual cooperation in 83 out of 100 runs. Although, in at least one generation, all genomes in 35 out of 83 runs were R -preferring, i.e., $u(R) > u(T)$, there were only five runs in which the best genome was R -preferring at the end. Even more surprisingly, R -preferring genomes were in the minority in all generations of 19 runs.

It was also shown that mutual cooperation followed a phase transition. In most runs, the phase transition period was less than 100 generations. The average numbers of R -preferring genomes at the phase transition suggested that about a half of the phase transition might occur even when R -preferring genomes were in the minority. We also

investigated the relation between the evolved u and Formula 2, which was originated in the previous paper [8], so as to know whether the evolved u could maintain the mutual cooperation by themselves. The result showed that most of all genomes in most of all runs satisfied the formula regardless of the phase transition. It may be a reason why mutual cooperation occurred even in the minority case.

Until now, this work is in a preliminary phase of showing results in a certain setting. We have to discuss the reason why such the results emerge and also have to investigate the effect of settings on the results. Especially, we have to clarify the property of the phase transition rigorously. Also, we should analyze the dynamics of the evolution process to elucidate the results of this paper.

7. REFERENCES

- [1] R. Axelrod. *The Evolution of Cooperation*. Basic Books, New York, 1984.
- [2] R. Axelrod. The Evolution of Strategies in the Iterated Prisoner’s Dilemma. In L. Davis, editor, *Genetic Algorithms and Simulated Annealing*, pages 32–41. Pitman / Morgan Kaufmann, London / Los Altos, CA, 1987.
- [3] E. Fehr and K. M. Schmidt. A Theory of Fairness, Competition, and Cooperation. *Quarterly Journal of Economics*, 114:817–868, 1999.
- [4] D. B. Fogel. Evolving behaviors in the iterated prisoner’s dilemma. *Evolutionary Computation*, 1:77–97, 1993.
- [5] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, Reading, MA, 1989.
- [6] P. Hingston and G. Kendall. Learning versus Evolution in Iterated Prisoner’s Dilemma. In *Proc. 2004 Congress on Evolutionary Computation, CEC’04*, pages 364–372, Portland, Oregon, U.S.A., 2004.
- [7] M. Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, Cambridge, MA, 1996.
- [8] K. Moriyama. Utility based Q-learning to facilitate cooperation in Prisoner’s Dilemma games. *Web Intelligence and Agent Systems*, 7(3):233–242, 2009.
- [9] W. Poundstone. *Prisoner’s Dilemma*. Doubleday, New York, 1992.
- [10] H.-Y. Quek, K. C. Tan, C.-K. Goh, and H. A. Abbass. Evolution and Incremental Learning in the Iterated Prisoner’s Dilemma. *IEEE Transactions on Evolutionary Computation*, 13:303–320, 2009.
- [11] J. K. Rilling, D. A. Gutman, T. R. Zeh, G. Pagnoni, G. S. Berns, and C. D. Kilts. A Neural Basis for Social Cooperation. *Neuron*, 35:395–405, 2002.
- [12] T. W. Sandholm and R. H. Crites. Multiagent reinforcement learning in the Iterated Prisoner’s Dilemma. *BioSystems*, 37:147–166, 1996.
- [13] J. L. Stimpson, M. A. Goodrich, and L. C. Walters. Satisficing and Learning Cooperation in the Prisoner’s Dilemma. In *Proc. 17th International Joint Conferences on Artificial Intelligence, IJCAI-01*, pages 535–540, Seattle, Washington, U.S.A., 2001.
- [14] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

- [15] F. Vega-Redondo. Long-run cooperation in the one-shot Prisoner's Dilemma: A hierarchic evolutionary approach. *BioSystems*, 37:39–47, 1996.
- [16] C. J. C. H. Watkins and P. Dayan. Technical Note: Q-learning. *Machine Learning*, 8:279–292, 1992.

Cooperation through Reciprocity in Multiagent Systems: An Evolutionary Analysis

Christian Hütter and Klemens Böhm
Institute for Program Structures and Data Organization
Karlsruhe Institute of Technology, Germany
[christian.huetter, klemens.boehm]@kit.edu

ABSTRACT

The Service Game is a model for reciprocity in multiagent systems. Here, agents interact repeatedly by requesting and providing services. In contrast to existing models where players are matched randomly, players of the Service Game may choose with whom they play. The rationale behind *provider selection* is to choose a provider that is likely to perform a task as desired. We develop a formal model for provider selection in the Service Game. An evolutionary process based on a genetic algorithm allows us to incorporate notions of bounded rationality, learning, and adaptation into the analysis of the game. We conduct a series of experiments to study the evolution of strategies and the emergence of cooperation. We show that cooperation is more expensive with provider selection than with random matching. Further, populations consisting of discriminators and defectors form a bistable community.

Categories and Subject Descriptors

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

General Terms

Economics, Experimentation

Keywords

Agent cooperation, Formal model, Social simulation, Evolution

1. INTRODUCTION

In settings without payments between individuals, the social mechanism of *reciprocity* is the basis for cooperation [16]. Examples for such settings are peer-to-peer systems and social search. With direct reciprocity, an individual A rewards helpful acts or punishes uncooperative acts of another individual B . Think of tit-for-tat, where A reciprocates B 's previous action. With indirect reciprocity in turn, an action is rewarded or punished by a third individual C , not involved in the original interaction. Such strategies typically rely on reputation and status [2]. While cooperation

Cite as: Cooperation through Reciprocity in Multiagent Systems: An Evolutionary Analysis, Christian Hütter and Klemens Böhm, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 241-248.
Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

through direct reciprocity can only emerge in repeated interactions between two individuals, it can emerge through indirect reciprocity in one-shot interactions as well.

Reciprocity has been a prominent research topic, both in theoretical [3, 16] and in experimental work [8, 10, 12]. In existing models for reciprocity, players are either matched randomly (for indirect reciprocity), or the same pairs of individuals interact repeatedly (for direct reciprocity). In realistic settings, however, individuals can choose whom to interact with. In the following, we will call this choice *provider selection*. Given the choice of interaction, both direct and indirect reciprocity can emerge. If individual A has done individual B a favor, A can redeem this favor either directly by interacting with B again or indirectly by interacting with another individual C . Thus, comprehensive models should cover both forms of reciprocity.

In this paper, we study the following research question: How efficient is reciprocity under provider selection? Answering this question is difficult for several reasons. First, we expect provider selection to change the game substantially, compared to random matching. This is because cooperative providers, which are typically preferred by requesters, are likely to have high workloads. Second, provider selection has not yet been analyzed in existing studies of reciprocity. As we will explain, a formal model required for the analysis cannot be derived directly from existing models where players are matched randomly. Third, an analytical solution of evolutionary games is hard if discriminating strategies are present [2]. Discriminating strategies differentiate between players according to certain attributes (e.g., their cooperativeness), as opposed to strategies that treat all players as equal. Simulations have shown that discriminating strategies can lead to cooperative populations [12].

We meet these challenges by proposing the *Service Game*, a formal model for reciprocity under provider selection. Here, players interact repeatedly in the roles of requester and provider. The *requester* sends a request to the *provider*, who decides on processing the request. Players have a benefit if other players process their requests, whereas processing requests for others incurs cost. In contrast to existing models, ours allows for both direct and indirect reciprocity by letting players choose with whom they play. Note that, in our model, all nodes are fully connected. Hence, network formation is not an issue here.

Traditional game theory assumes that all players are fully rational and homogeneous. While these assumptions are crucial for the mathematical tractability of the models, more recent work has dropped them [14]. In evolutionary game

theory, the role of perfect rationality is taken over by a learning process in a heterogeneous society of agents. An evolutionary process modeled by a genetic algorithm updates the behavior of the agents. Genetic algorithms have frequently been used in experimental work on cooperation to describe social learning [1, 10, 14].

We have designed and carried out a series of experiments to study the evolution of strategies and the emergence of cooperation under provider selection. It shows that cooperation is more expensive with provider selection than with random matching. We deem this a fundamental observation because it means that direct and indirect reciprocity should not be analyzed separately. Further, populations consisting of discriminators and defectors form a bistable community. That is, provider selection preserves the evolutionary stable states predicted by theory [3].

2. RELATED WORK

In this section we discuss related work. After examining existing models for reciprocity, we review studies of the evolution of cooperation. Finally, we delimit our work from existing studies of provider selection.

While direct reciprocity has been a popular research topic, more recent work on reciprocity has focused on indirect reciprocity. One popular model to analyze cooperation through indirect reciprocity is the *helping game* [12]. There, pairs of a donor and a recipient are formed randomly. The recipient asks the donor he is matched with for costly ‘help’. Nowak and Sigmund have shown that cooperation can evolve if information about the past behavior of the players is available. Each player is assigned a public *image score* which increases through cooperation and decreases through defection. Strategies based on image scoring have proven to be evolutionary stable. A strategy population is called *evolutionary stable* if the evolutionary process rejects the invasion of the population by mutant strategies [9].

Gal and Pfeffer [5] showed that reciprocity has significant implications for the behavior of agents. When they interact with humans over time, agents need to learn social factors that affect people’s play. Brandt et al. [3] analyzed direct and indirect reciprocity theoretically using replicator dynamics. While replicator dynamics are common to study the evolutionary dynamics in games, they lack the notions of learning and adaption [14]. Researchers often resort to numerical simulations using genetic algorithms [8, 10, 12], which explicitly model the individual agents. We will make use of this methodology as well by carrying out an evolutionary analysis of cooperation in multiagent systems. Research has proposed various approaches on cooperation among deceptive agents that misreport the behavior of other agents. Sen [15] has proposed a probabilistic strategy based on reciprocity which is able to resist this kind of deception. While we do not consider deceptive agents, the approach by Sen might be applied to provider selection as well.

There also are models for the evolution of cooperation not based on reciprocity. One such model uses so-called ‘tags’ [6] to help agents in identifying the groups they belong to. It has been shown that provider selection based on tags produces stable cooperation. While tags correspond to the origin of the agents, provider selection in our setting is based on behavior. Fullam et al. [4] proposed a testbed for trust and reputation mechanisms. Agents may choose other agents from which they seek help, and the agents asked for

help decide whether to cooperate or not. Whereas Fullam et al. compare a fixed set of strategies, we investigate the evolution of strategies through a genetic algorithm. Provider selection has also been studied in the area of peer-to-peer systems [13]. However, existing studies we are aware of focus on performance figures such as the ratio of successful interactions. In the following, we develop a full-fledged economic model for reciprocity in multiagent systems.

3. THE SERVICE GAME

The Service Game is a repeated game where players may send requests to other players and decide whose requests they process. That is, each player actually plays multiple games in one round—one as requester and zero or more as provider, depending on the number of requests he receives. The *fee* for sending a request to another player is f , with $f > 0$. The *cost* of processing the request of another player is c , with $c > 0$. The *benefit* of a player if another player processes his request is b , with $b > 0$. Fee, cost and benefit are the same for all players. As usual in helping games, we assume that $b > c > f > 0$ holds. The Service Game models a homogeneous system in which all players send and process requests at the same rate. To simplify matters, we assume that there is only one kind of service which every player can provide. Providers may neither pass on requests to other players, nor postpone the decision on processing a request. All players make their cooperation decisions simultaneously.

We implement the Service Game as a multiagent system, i.e., agents act as players. Each agent pursues two so called ‘policies’ which define the actions it performs. A *placing policy* specifies the set of players which the agent sends requests to. An *accepting policy* specifies the set of players whose requests the agent is willing to process. As described in Algorithm 1, every round t of the game consists of two steps: provider selection and service decisions. First, each agent i evaluates its placing policy p_i to select a provider j . If the placing policy is empty, the agent does not send a request. If it contains several players, a provider is selected at random from this set. Second, each agent j decides on processing incoming requests by evaluating its accepting policy a_j . If the accepting policy contains requester i , the agent processes the task, otherwise it rejects the request.

Algorithm 1 The Service Game

```

for all rounds  $t$  do
  for all players  $i \in I$  do {provider selection}
    provider  $j \leftarrow \text{evaluate}(p_i)$ 
    queue  $Q_j \leftarrow Q_j \cup \{i\}$ 
  for all players  $j \in I$  do {service decisions}
    for all requesters  $i \in Q_j$  do
      if  $i \in \text{evaluate}(a_j)$  then
        process the task
      else
        reject the request

```

3.1 Formal Model

The Service Game is characterized by the set of players $I = \{1, \dots, n\}$, the policy sets A and P , and the payoff or net return n_k^t of player $k \in I$ in round t of the game. A *placing policy* $p_k \subseteq I \setminus \{k\}$ specifies the set of players which player k sends a request to. E.g., the placing policy $p_k =$

$\{l \in I \setminus \{k\} \mid r_l \geq 0.8\}$ means that player k sends requests only to players with a cooperativeness of 80% or more. An *accepting policy* $a_k \subseteq I \setminus \{k\}$ specifies the set of players whose requests player k is willing to process. E.g., the accepting policy $a_k = \{l \in I \setminus \{k\} \mid r_l \geq 0.5\}$ means that player k processes requests only for players with a cooperativeness of 50% or more. We denote the set of placing policies as $P = \{p_1, \dots, p_n\}$ and the set of accepting policies as $A = \{a_1, \dots, a_n\}$. We say that player k *meets* placing policy p resp. accepting policy a if $k \in p$ resp. $k \in a$ holds.

We introduce a *discount rate* $\delta \leq 1$ to avoid infinite payoffs. On the one hand, it takes into account the probability that another round occurs after the current round. On the other hand, it represents the economic fact that the current value of a future income is less than its nominal value. The *present value* N_k^t of the payoffs in round t is the discounted sum of all previous rounds.

$$N_k^t = \delta \cdot N_k^{t-1} + n_k^t \quad (1)$$

3.2 Cooperativeness of the Players

Number of requests received

We first count how many requests player k is expected to receive in round t . Let $|p_l^t|$ be the number of players who meet the placing policy p_l^t of some player l in round t . The reciprocal value $1/|p_l^t|$ then is the probability that a random player who meets the placing policy p_l^t receives a request from player l . We denote this probability as the expected number of requests exp_l^t such a player receives from player l in round t .

$$exp_l^t = \begin{cases} \frac{1}{|p_l^t|} & \text{if } |p_l^t| > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (2)$$

We now sum up exp_l^t over all players l from which player k might receive requests because he meets their placing policies p_l^t . This is the expected number of requests rec_k^t player k receives in round t .

$$rec_k^t = \sum_{l \in \{I \setminus \{k\} \mid k \in p_l^t\}} exp_l^t \quad (3)$$

Number of requests processed

Next, we count how many requests player k is expected to process in round t . He accepts requests from player l if and only if this player meets his accepting policy a_k^t . We refer to the respective indicator function as $acc_{k,l}^t$.

$$acc_{k,l}^t = \begin{cases} 1 & \text{if } l \in a_k^t, \\ 0 & \text{otherwise.} \end{cases} \quad (4)$$

If we multiply $acc_{k,l}^t$ by exp_l^t , we get the expected number of requests which player k processes for player l . We now sum up this product over all players l from which player k receives requests because he meets their placing policies p_l^t . This is the expected number of requests $proc_k^t$ player k processes.

$$proc_k^t = \sum_{l \in \{I \setminus \{k\} \mid k \in p_l^t\}} acc_{k,l}^t \cdot exp_l^t \quad (5)$$

Cooperativeness

We define the *cooperativeness* of a player as the share of incoming requests he processes. E.g., a player with cooperativeness $r = 0.5$ processes half of the requests. The cooperativeness r_k^t of player k in round t is the quotient of

the number of requests processed $proc_k^t$ and the number of requests received rec_k^t .

$$r_k^t = \begin{cases} \frac{proc_k^t}{rec_k^t} & \text{if } rec_k^t > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (6)$$

The *mean cooperativeness* R_k^t of player k in round t is the average over all previous rounds. Since players may change their behavior during the game, recent interactions are more important than old ones. We use an exponential moving average with smoothing factor $\alpha \leq 1$ to assign more weight to recent interactions.

$$R_k^t = \alpha \cdot r_k^t + (1 - \alpha) \cdot R_k^{t-1} \quad (7)$$

3.3 Payoff of the Players

Benefit of a successful request

To compute the benefit b_k^t of player k , we first determine whether this player sends out a request in round t . Player k sends requests to all players l which meet his placing policy p_k^t . We refer to the respective indicator function as $sent_k^t$.

$$sent_k^t = \begin{cases} 1 & \text{if } |p_k^t| > 0, \\ 0 & \text{otherwise.} \end{cases} \quad (8)$$

Next, we determine whether this request is successful or not. Player l accepts requests from player k ($acc_{l,k}^t = 1$) if and only if player k meets his accepting policy a_l^t . If we multiply $acc_{l,k}^t$ by exp_k^t , we get the expected number of requests which player l processes for player k . We now sum up this product over all players l to which player k sends requests because they meet his placing policy p_k^t .

$$succ_k^t = \sum_{l \in p_k^t} acc_{l,k}^t \cdot exp_k^t \quad (9)$$

Note that the expected number of successful requests $succ_k^t$ is at most 1. The benefit b_k^t of player k is the product of b and the number of successful requests sent out in round t .

$$b_k^t = b \cdot succ_k^t \quad (10)$$

Cost of processing requests

The cost c_k^t depends on the number of requests player k processes in round t . While player k receives requests from all players l whose placing policies p_l^t he meets, he only processes requests from players who meet his accepting policy a_k^t . This is exactly the definition of $proc_k^t$ in Equation (5). In addition, player k has to pay fee f if he sends out a request in round t . We use the indicator function $sent_k^t$ from Equation (8).

$$c_k^t = c \cdot proc_k^t + f \cdot sent_k^t \quad (11)$$

Payoff

The payoff n_k^t of each player k in round t is the difference between the benefit b_k^t of a successful request and the cost c_k^t of sending and processing requests.

$$n_k^t = b \cdot succ_k^t - c \cdot proc_k^t - f \cdot sent_k^t \quad (12)$$

In every round t of the game, each player k would ideally choose the combination of placing policy p_k^t and accepting

policy a_k^t which maximizes his payoff. We denote the optimal payoff of player k in round t as \hat{n}_k^t :

$$\hat{n}_k^t = \underset{p_k^t \subseteq I \setminus \{k\}, a_k^t \subseteq I \setminus \{k\}}{\text{maximize}} n_k^t \quad (13)$$

It is hard to determine the optimal payoff analytically if discriminating strategies are present [2]. Often, pairs of discriminating strategies perform equally well against each other, so that their frequencies drift randomly. However, the success of other strategies depends on the frequencies of the discriminating strategies. To remove this restriction, we will resort to numerical simulations.

4. EVOLUTIONARY ANALYSIS

In evolutionary games, players are not assumed to be rational or able to think ahead. Strategies are simple behavioral programs which specify the actions of each player. In this section, we will first explain the strategies used in our analysis and then describe the evolutionary process in detail.

4.1 Strategies

In a previous experiment of ours [7], subjects have formulated plaintext policies which we then used to program agents playing the Service Game. A *cutoff strategy* based on the cooperativeness of the players was the most successful strategy for provider selection. Hence, in this paper, we will analyze the evolutionary stability of the cutoff strategy for provider selection. In each round t of the game, player i sends a request to player j if the mean cooperativeness R_j^{t-1} of player j exceeds a threshold value ρ_i :

$$p_{i,j}^t = \begin{cases} 1 & \text{if } R_j^{t-1} \geq \rho_i, \\ 0 & \text{otherwise.} \end{cases} \quad (14)$$

We will eventually compare the results of our study with previous studies on reciprocity [2]. In these studies, the three basic strategies ‘always cooperate’, ‘always defect’, and ‘tit-for-tat’ have prevailed. Thus, in this paper, we use these strategies as well for service decisions. Needless to say, there are many other strategies conceivable for both provider selection and service decisions. Nevertheless, the strategies we use capture the most important aspects of cooperation [3].

We implement the strategies for service decisions as *stochastic reactive strategies* [11]. They are described by two parameters (p, q) , which are the probabilities to cooperate after a cooperation resp. a defection by the co-player in the previous round. Always cooperate is given by $(1, 1)$, always defect by $(0, 0)$, and tit-for-tat by $(1, 0)$. In each round t , the probability that player i accepts a request from player j is the expected value of the stochastic reactive strategy. Recall that $a_{j,i}^{t-1}$ denotes the service decision of player j regarding player i in round $t - 1$.

$$a_{i,j}^t = p \cdot a_{j,i}^{t-1} + q \cdot (1 - a_{j,i}^{t-1}) \quad (15)$$

A strategy can be represented as a ‘chromosome’ describing the action of a player in each different context of the game. We encode the strategy of each player i by a binary string of 24 bits. The first 8 bits represent the threshold value ρ_i of the cutoff strategy for provider selection. The second and third 8 bits represent the probabilities p_i and q_i of the stochastic reactive strategy for service decisions. We use a Gray coding of the binary strings to avoid the representational bias in binary encoding.

4.2 Evolutionary Process

We update the strategies through an evolutionary process modeled by a genetic algorithm (GA). GAs mimic a population of strategies acting in a well-defined environment which evaluates the performance of each strategy. New populations are formed by selecting the better performing strategies and modifying them through genetic operators. The GA then subjects the resulting offspring to competition with other strategies in the population. Successful strategies are allowed to reproduce, while unsuccessful strategies become extinct. GAs have frequently been used in economics to characterize a form of social learning [1, 10, 14]. Selection can be interpreted as learning by imitation, recombination as learning by communication, and mutation as learning by experiment. The combination of these three operators results in a very powerful optimization algorithm.

Algorithm 2 The genetic algorithm

```

Create initial population  $m_0$ 
Evaluate  $m_0$ 
for round  $t = 1$  to  $t_{max}$  do
    Rank  $m_{t-1}$ 
    Select from  $m_{t-1}$  into  $m_t$ 
    Recombine  $m_t$ 
    Mutate  $m_t$ 
    Evaluate  $m_t$ 

```

The GA used in this paper is shown in Algorithm 2. First, an initial strategy population m_0 is created randomly. Each strategy is then tested against the environment (composed of the other strategies) and receives a performance score (the payoff). Given the performance scores, a fitness value is assigned to each strategy. We use a rank-based fitness function with a selective pressure of 2 and linear ranking, giving the most fit strategy a fitness value of 2 and the least fit strategy a fitness value of 0. Strategies from the parent population m_{t-1} are selected for reproduction using a stochastic universal sampling routine. The strategies m_t selected for reproduction are recombined using a single-point crossover function with probability .7. Having produced the offspring, mutation may now be applied with probability $.7/L_{ind}$, where $L_{ind} = 24$ is the length of a chromosome. Finally, the offspring m_t is evaluated and the new performance scores are calculated. The GA terminates when the maximum number of rounds t_{max} is reached. We verify that the population has converged by analyzing its variance.

5. EXPERIMENTS

To analyze the evolution of strategies and the emergence of cooperation under provider selection, we formulate the following research questions:

1. How does provider selection change policy-based helping scenarios?
2. Which states of the game are evolutionary stable, and how efficient are the resulting equilibria?
3. How do the parameters of the game influence stability and efficiency?

Regarding the first question, we expect provider selection to change the game substantially. This is because cooperative providers are likely to receive many requests, making cooperation expensive. With the second question we investigate whether the equilibria identified in previous studies

Table 1: Parameter values used in the experiments

Experiments		Parameters				
		f	c	b	δ	α
$E1$	Random matching	0	2	20	.9	.5
	Provider selection	0	2	20	.9	.5
$E2$	No fee	0	2	20	.9	.5
	Low cost	1	2	20	.9	.5
	High cost	5	10	20	.9	.5
$E3$	Low discount	1	2	20	.1	.5
	Medium discount	1	2	20	.5	.5
	High discount	1	2	20	.9	.5
$E4$	Low smoothing	1	2	20	.5	.1
	Medium smoothing	1	2	20	.5	.5
	High smoothing	1	2	20	.5	.9

still hold for the Service Game. We expect the equilibria to be less efficient because, according to a previous study of ours [7], players are less cooperative in a game with provider selection than in a game with random matching. The third question addresses the parameters of the Service Game (f , c , b , δ , α). According to previous studies on reciprocity [2], we expect the cost-to-benefit ratio c/b to have a significant influence on the efficiency of the game. While a high discount factor δ causes a low rate of inflation, a low smoothing factor α indicates a long memory of interactions. We expect both to increase the efficiency.

5.1 Experimental Design

To answer the research questions, we have designed and conducted a series of experiments. In these experiments, we explore all parameters of the game separately (i.e., we vary one parameter at a time). In Experiment $E1$, we compare random matching with provider selection, to address the first research question. We tackle the second question by analyzing the equilibria of the game and their efficiency. Regarding the third question, we compare different ratios of fee/cost and cost/benefit in Experiment $E2$, different discount rates in Experiment $E3$, and different smoothing factors in Experiment $E4$. Table 1 serves as a summary. Finally, we identify lessons learned from our experiments that are of general interest.

Each simulation consists of a population of 100 agents, which played the Service Game for 100 rounds. In each round of the game, each player plays against every other player specified by his placing policy. The payoff then is the expected value of the play, i.e., the average over the individual games. Payoffs were calculated using the equations in Section 3.3. Under each of the conditions, 100 repetitions were conducted to allow for stochastic variations. The choices of simulation parameters (e.g., population size) and algorithmic components (e.g., selection function of the GA) were guided by considerations of robustness and computational constraints. Note that genetic algorithms are extremely robust to actual parametric and algorithmic choices. All the results reported in the next section have been confirmed using a variety of different simulation parameters and algorithmic components.

5.2 Methodology

The frequencies of the three strategies for service decisions (cooperate, defect, and tit-for-tat) are given by x , y ,

and z with $x + y + z = 1$. Thus, the three strategies form a *strategy simplex* which describes the composition of the population. The strategy simplex can be imagined as a 2D triangle in the 3D coordinate system (x, y, z) . The three vertices $x = 1$, $y = 1$, and $z = 1$ of the strategy simplex describe ‘pure’ populations consisting solely of the strategies cooperate, defect, and tit-for-tat, respectively. The edges $x = 0$, $y = 0$, and $z = 0$ describe ‘dual’ populations consisting of the two strategies specified by its vertices. Finally, the points (x, y, z) within the simplex describe ‘mixed’ strategy populations.

By definition, the number of points in the strategy simplex is infinite. Thus, we have to create a random sample of the strategy simplex. In theory, it is possible that the random sample misses important aspects of the game. In the following experiments, we have chosen a large sample size of 500 points to minimize the possibility of error. To avoid clutter in the figure, we will depict only 100 points. We run the GA for each point of the sample and analyze how the population evolves. The strategies are updated through the evolutionary process described in Section 4. In the following analyses, we will average over the resulting strategies in each population. The key figures to quantify the efficiency are the mean cooperativeness R and the total payoff N of the players. Both figures are strongly correlated because requesters can only make profit if the providers cooperate. We focus on the cooperativeness because it is, by definition, normalized to the interval $[0, 1]$ and thus allows for a direct comparison.

6. RESULTS

In this section, we first examine a system with random matching to replicate the effects of existing studies [2] on reciprocity. It serves as baseline for the following analysis of provider selection. Our experiments indicate that there are significant differences between random matching and provider selection.

6.1 Random Matching

In experiment $E1$, we have analyzed how the matching of the players affects the evolution of the strategies and the efficiency of the game. Figure 1a shows the resulting simplex for random matching. To visualize the results, we have projected the strategy simplex onto a 2D coordinate system. The three vertices $x = 1$, $y = 1$, and $z = 1$ were mapped to the coordinates $(1, 0)$, $(0, 0)$, and $(1/2, \sqrt{3}/2)$, respectively. Recall that each point of the simplex represents one population of strategies. The coordinates specify the frequencies of the strategies, i.e., the initial composition of the population. To visualize the efficiency of each population, we depict the average cooperativeness as the color of the corresponding point. Light colors (white to yellow) represent cooperative populations, dark colors (red to black) uncooperative ones.

During the evolutionary process, the composition of the population changes because successful strategies prevail. The arrows in each point visualize the direction which the population shifts to. E.g., the points in the center of the simplex shift to the edge $y = 0$. A strategy population is in an evolutionary stable state if its composition does not change during the evolutionary process [9]. A fixed point keeps its position, i.e., the population is evolutionary stable. A closer look at the data reveals that the vertex $y = 1$ consisting solely of defectors is evolutionary stable. Furthermore, all points on the

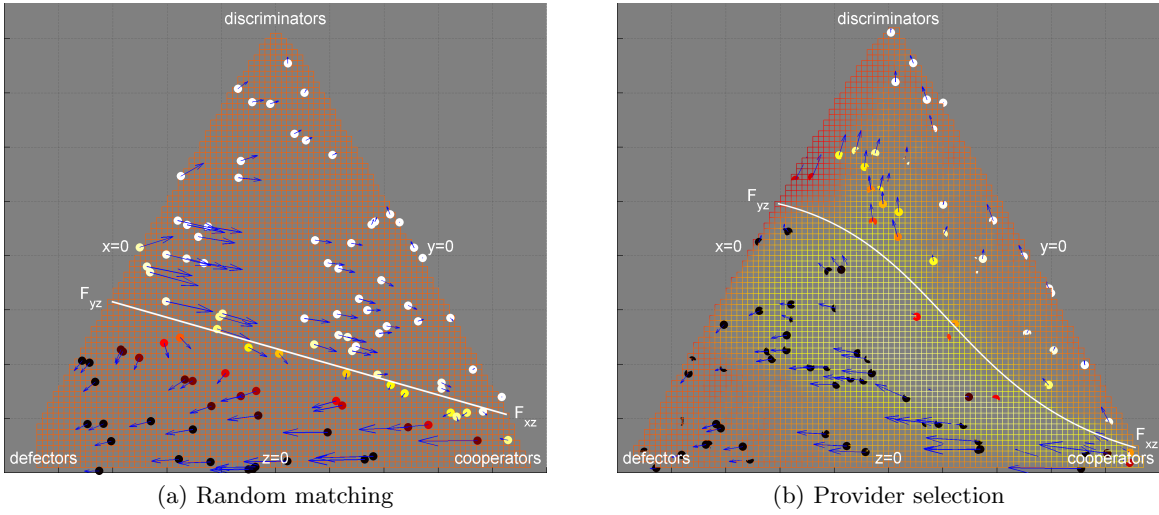


Figure 1: Strategy simplices for random matching and for provider selection.

edge $y = 0$ are evolutionary stable. I.e., any mixture of cooperators and discriminators is in equilibrium. All points on the edge $z = 0$ are unstable. There is a line of unstable points connecting a fixed point F_{yz} on the edge $x = 0$ with another fixed point F_{xz} on the edge $y = 0$. We call this line the *boundary line*. While the states between the defectors vertex and the boundary line are inefficient (cooperativeness $R \approx 0$), the states between the line and the discriminators vertex are efficient ($R \approx 1$). These findings are consistent with theoretical results in [3].

6.2 Provider Selection

From now on, we focus on systems with provider selection. Figure 1b shows the resulting strategy simplex. Apparently, the system with random matching has a higher fraction of cooperative (white) states than the system with provider selection. In fact, the mean cooperativeness is $.637 (\pm .431)$ for random matching and only $.402 (\pm .435)$ for provider selection. A t-test confirms that the difference is significant for a confidence level of $.01$.

Observation 1. Games with provider selection are less cooperative than games with random matching.

This result is expected because, according to their placing policies, requesters prefer cooperative providers. Thus, cooperative players have high workloads, making cooperation expensive. In contrast, random matching balances requests uniformly between providers.

An analysis of the evolutionary stability shows that the system with provider selection has the same stable states as the system with random matching. Again, the vertex $y = 1$ consisting solely of defectors is evolutionary stable. Along the edge $y = 0$, any mixture of cooperators and discriminators is stable. Finally, a line of unstable states (the boundary line) connects a fixed point F_{yz} on the edge $x = 0$ with a fixed point F_{xz} on the edge $y = 0$.

Observation 2. Populations consisting of discriminators and defectors form a bistable community.

While the strategies for service decisions are defined by their coordinates in the strategy simplex, the cutoff strategy for provider selection is defined by its threshold values ρ

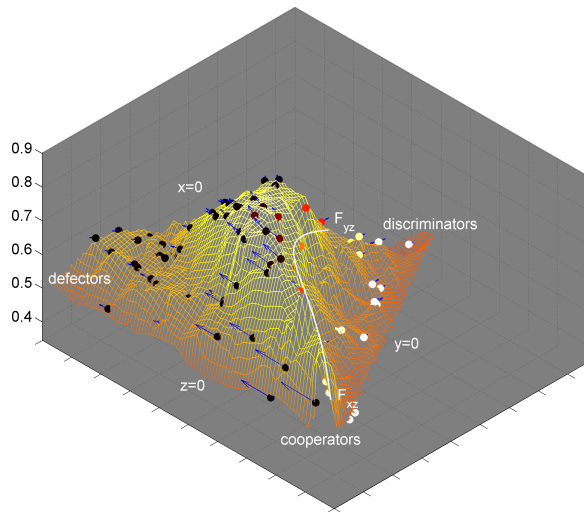
(see Equation 14). For each population of strategies, we have computed the mean and the variance of the threshold values. We visualize the average threshold values as an additional dimension (the z -axis) of the strategy simplex. ‘Higher’ points represent greater threshold values and thus stricter cutoff strategies. Figure 2 shows the resulting strategy simplices.

Interestingly, the location of the boundary line strongly influences the threshold values of the cutoff strategies (height). The boundary line divides the simplex into two areas. In the area between the defectors vertex and the boundary line, the cutoff thresholds reach their maximum values, and the system is inefficient ($R \approx 0$). The threshold maximum can be visualized as a ‘range’ of strict cutoff strategies. On the other side, between the boundary line and the discriminators vertex, the threshold values are much lower. Visually, the boundary line forms the ‘rim’ of the cutoff range. If we descend the cutoff range towards the discriminators vertex, the system becomes more efficient. In the ‘valley’ along the edge $y = 0$, the system is highly efficient ($R \approx 1$).

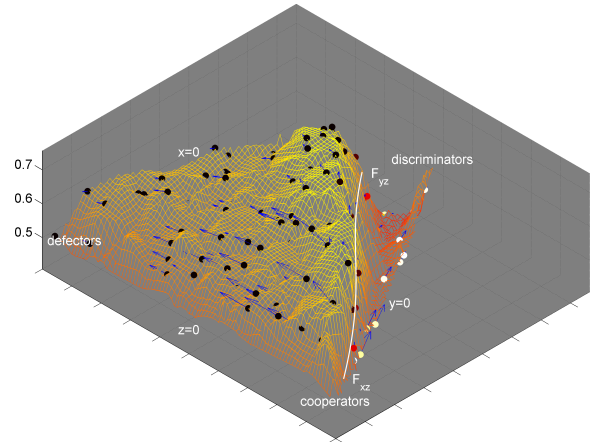
6.2.1 Benefit, Cost, and Fee

In experiment *E2*, we have investigated how the parameters benefit, cost, and fee affect the efficiency. First, we have conducted an analysis of variance (ANOVA) to test whether the means of the three treatments (no fee, low cost, high cost) are all equal or not. The test indicates that at least one sample mean is different from the other two ($F = 60.89$) with a significance level of $.01$. Next, we have analyzed the ratio f/c of fee to cost by comparing the no-fee and the low-cost treatment. The statistics do not show any significant difference. Thus, we leave aside the no-fee treatment in the remaining analysis.

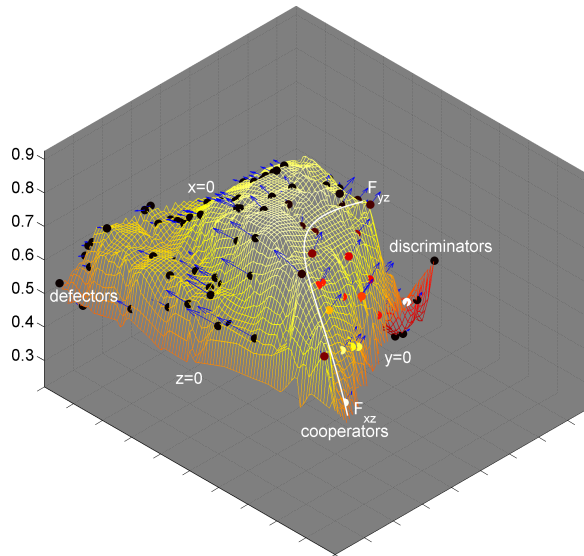
The ratio c/b of cost to benefit can be analyzed by comparing the low-cost and the high-cost treatment. Figures 2a–b show the resulting strategy simplices for both treatments. The mean cooperativeness is $.37 (\pm .43)$ for the low-cost and only $.148 (\pm .316)$ for the high-cost treatment. A t-test confirms that the difference is significant for a confidence level of $.01$. We observe that the ratio c/b also affects the location of the boundary line. As the cost-to-benefit ratio increases,



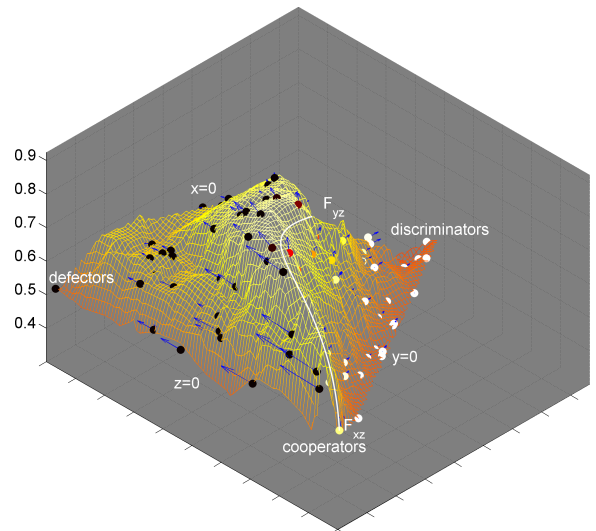
(a) Low cost/high discount: $f=1, c=2, b=20, \delta=.9$



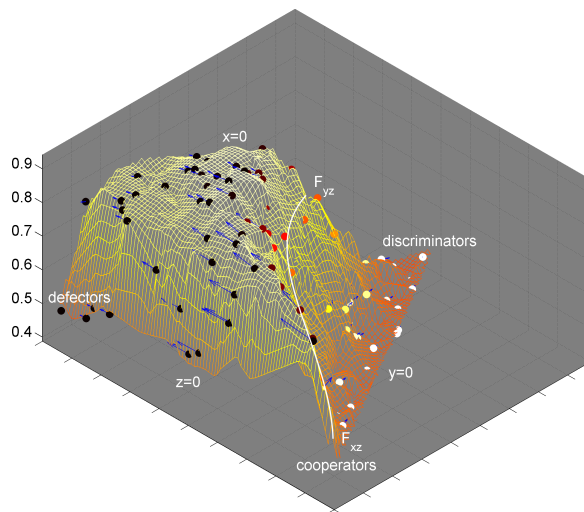
(b) High cost: $f=5, c=10, b=20$



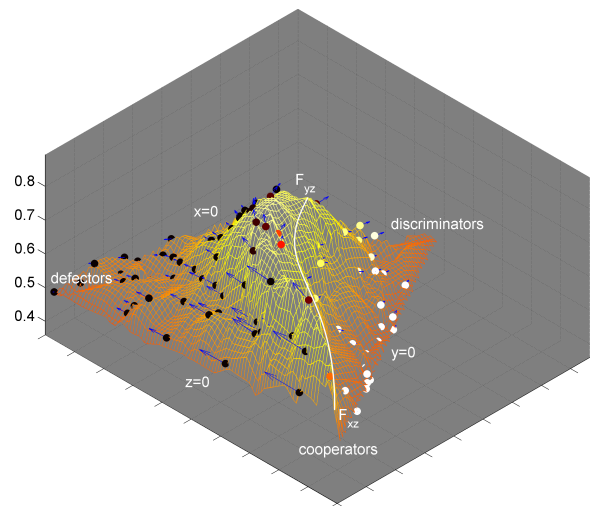
(c) Low discount rate: $\delta=.1, \alpha=.5$



(d) Medium discount/smoothing: $\delta=.5, \alpha=.5$



(e) Low smoothing factor: $\delta=.5, \alpha=.1$



(f) High smoothing factor: $\delta=.5, \alpha=.9$

Figure 2: Strategy simplices for different game parameters under provider selection.

the fixed point F_{yz} moves closer towards the discriminators vertex and thus the efficiency decreases.

Observation 3. The lower the ratio of cost to benefit, the more efficient the system.

An important lesson learned is that the benefit of a successful request should be at least one order of magnitude higher than the cost of processing. A fee for sending requests does not affect the efficiency of the system.

6.2.2 Discount Rate and Smoothing Factor

In experiments $E3$ and $E4$, we have investigated how discount rate and smoothing factor affect the efficiency. First, we have compared low, medium and high discount rates δ . A low discount rate indicates a high rate of inflation and thus a decline of the present value of the payoffs. For the results of low, medium, and high discount rates see Figures 2c, d, a. The mean cooperativeness is .092 ($\pm .213$) for the low discount and .370 ($\pm .43$) for the high discount treatment. A t-test confirms that the difference is significant for a confidence level of .01. Apparently, the discount rate affects the ‘height’ of the cutoff range. The lower the discount rate, the higher the cutoff range and thus the stricter the strategies.

Observation 4. A higher discount rate (i.e., a lower inflation) results in a more efficient system.

Finally, we compared low, medium, and high smoothing factors α . A higher factor assigns more weight to recent interactions and thus causes a shorter memory. For the results of low, medium, and high smoothing factors see Figures 2e, d, f. The mean cooperativeness is .422 ($\pm .412$) for the low smoothing and .362 ($\pm .429$) for the high smoothing treatment. A t-test confirms that the difference is significant for a confidence level of .01. Apparently, the smoothing factor affects the ‘width’ of the cutoff range. The lower the factor is, the wider the cutoff range.

Observation 5. A lower smoothing factor (i.e., a longer memory) increases the efficiency of the system.

Thus, a second lesson learned is that reputation systems for multiagent systems should consider the complete history of interactions.

7. CONCLUSIONS

In settings without payments between individuals, reciprocity is the basis for cooperation. Examples for such settings are peer-to-peer systems and social search. In existing models for reciprocity, individuals are either matched randomly, or the same pairs of individuals interact repeatedly. However, in realistic settings, individuals can choose whom to interact with. In this paper, we have investigated how efficient reciprocity is under provider selection. To do so, we have developed a formal model for reciprocity in multiagent systems. Strategies are updated through an evolutionary process based on a genetic algorithm. This lets us incorporate the notions of bounded rationality, learning, and adaptation into the analysis.

We have designed and carried out a series of experiments to study the evolution of strategies and the emergence of cooperation. Our results show that cooperation is more expensive in a system with provider selection than in a system with random matching. Thus, existing models for

reciprocity overestimate the efficiency of real-world systems where both direct and indirect reciprocity may occur in combination. Further, populations consisting of discriminators and defectors form a bistable community.

8. ACKNOWLEDGMENTS

The research leading to these results has received funding from the European Community’s Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 216287 (TAS³ – Trusted Architecture for Securely Shared Services).

9. REFERENCES

- [1] R. Axelrod. The evolution of strategies in the iterated prisoner’s dilemma. In *Genetic Algorithms and Simulated Annealing*. Pitman, 1987.
- [2] H. Brandt, H. Ohtsuki, Y. Iwasa, and K. Sigmund. A survey of indirect reciprocity. *Mathematics for Ecology and Environmental Sciences*, 2007.
- [3] H. Brandt and K. Sigmund. The good, the bad and the discriminator—errors in direct and indirect reciprocity. *Journal of theoretical biology*, 239(2), 2006.
- [4] K. K. Fullam and K. S. Barber. Learning trust strategies in reputation exchange networks. In *Proc. of the 5th international joint conference on Autonomous agents and multiagent systems*, 2006.
- [5] Y. Gal and A. Pfeffer. Modeling reciprocal behavior in human bilateral negotiation. In *Proc. of the 22nd national conference on Artificial intelligence*, 2007.
- [6] D. Hales and B. Edmonds. Evolving Social Rationality for MAS using “Tags”. In *Proc. of the 2nd international joint conference on Autonomous agents and multiagent systems*, 2003.
- [7] C. Hütter, J. Z. Yue, C. von der Weth, and K. Böhm. Strategic provider selection in a policy-based helping scenario. In *Proc. of the 12th IEEE Conference on Commerce and Enterprise Computing*, 2010.
- [8] O. Leimar and P. Hammerstein. Evolution of cooperation through indirect reciprocity. *Biological sciences*, 268(1468), 2001.
- [9] J. Maynard Smith. *Evolution and the Theory of Games*. Cambridge Univ. Press, 1982.
- [10] J. Miller. The coevolution of automata in the repeated Prisoner’s Dilemma. *Journal of Economic Behavior & Organization*, 29(1), 1996.
- [11] M. Nowak and K. Sigmund. The evolution of stochastic strategies in the prisoner’s dilemma. *Acta Applicandae Mathematicae*, 20(3), 1990.
- [12] M. A. Nowak and K. Sigmund. Evolution of Indirect Reciprocity by Image Scoring. *Nature*, 393, 1998.
- [13] T. Papaioannou and G. Stamoulis. Effective use of reputation in peer-to-peer environments. In *Proc. of the IEEE International Symposium on Cluster Computing and the Grid*, 2004.
- [14] T. Riechmann. Genetic algorithm learning and evolutionary games. *Journal of Economic Dynamics and Control*, 25(6-7), 2001.
- [15] S. Sen. Believing others: Pros and cons. *Artificial Intelligence*, 142(2), 2002.
- [16] R. L. Trivers. The Evolution of Reciprocal Altruism. *The Quarterly Review of Biology*, 46(1), 1971.

Distributed Cooperation in Wireless Sensor Networks

Mihail Mihaylov
Vrije Universiteit Brussel
Pleinlaan 2
Brussels, Belgium
mmihaylo@vub.ac.be

Yann-Aël Le Borgne
Vrije Universiteit Brussel
Pleinlaan 2
Brussels, Belgium
yleborgn@vub.ac.be

Karl Tuyls
Maastricht University
Minderbroedersberg 6a
Maastricht, The Netherlands
k.tuyls@maastrichtuniversity.nl

Ann Nowé
Vrije Universiteit Brussel
Pleinlaan 2
Brussels, Belgium
ann.nowe@vub.ac.be

ABSTRACT

We present a game-theoretic self-organizing approach for scheduling the radio activity of wireless sensor nodes. Our approach makes each node play a *win-stay lose-shift* (WSLS) strategy to choose when to schedule radio transmission, reception and sleeping periods. The proposed strategy relies only on local interactions with neighboring nodes, and is thus fully decentralized. This behavior results in shorter communication schedules, allowing to not only reduce energy consumption by reducing the wake-up cycles of sensor nodes, but also to decrease the data retrieval latency. We implement this WSLS approach in the OMNeT++ sensor network simulator where nodes are organized in three topologies — line, grid and random. We compare the performance of our approach to two state-of-the-art scheduling protocols, namely S-MAC and D-MAC, and show that the WSLS strategy brings significant gains in terms of energy savings, while at the same time reduces communication delays. In addition, we show that our approach performs particularly well in large, random topologies.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*intelligent agents, multiagent systems, coherence and coordination*; C.2.1 [Computer Communication Networks]: Network Architecture and Design—*distributed networks, wireless communication*

General Terms

Algorithms, Performance

Keywords

multiagent learning, collective intelligence, teamwork, coalition formation, coordination, implicit cooperation, emergent behavior

Cite as: Distributed Cooperation in Wireless Sensor Networks, M. Mihaylov, Y-A. Le Borgne, K. Tuyls and A. Nowé, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 249–256.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

1. INTRODUCTION

Wireless Sensor Networks (WSNs) are a recent class of networks able to monitor our daily environment with a high spatiotemporal accuracy [10, 2]. WSNs are composed of small sensing devices, also known as wireless sensor nodes, endowed with sensing, processing and wireless communication capabilities. Given the current technological trend, WSNs are envisioned to be mass produced at low cost in the next decade, for applications in a wide variety of domains. These include, to name a few, ecology, industry, transportation, or defense [2].

A typical WSN scenario consists of a set of sensor nodes, scattered in an environment, which report their data periodically to a centralized entity called *base station*. The resources of the untethered sensor nodes are often strongly constrained, particularly in terms of energy and communication. The base station usually possesses much larger resources, comparable to those of a standard laptop or desktop computer [10, 2].

The limited resources of the sensor nodes make the design of a WSN application challenging. Application requirements, in terms of latency, data throughput, or lifetime, often conflict with the network capacity and energy resources. The standard approach for addressing these tradeoffs is to rely on *wake-up scheduling* [10], which consists in alternating the active and sleep states of sensor nodes. In the active state, all the components of a node (CPU, sensors, radio) are active, allowing the node to collect, process and communicate information. In the sleep state, all these components are switched off, allowing the node to run with an almost negligible amount of energy. However, nodes in sleep mode cannot communicate with others, since their radio transmitter is switched off. The fraction of time in which the node is in the active mode is referred to as *duty cycle* [19].

Wake-up scheduling offers an efficient way to significantly improve the lifetime of a WSN application, and is well illustrated by S-MAC, a standard synchronized medium access control (MAC) protocol for WSN [20]. In S-MAC, the duty-cycle is fixed by the user, and all sensor nodes synchronize in such a way that their active periods take place at the same time. This synchronized active period enables neighboring nodes to communicate with one another. The use of routing then allows any pair of node to exchange messages. By tuning the duty-cycle, wake-up scheduling therefore allows

to adapt the use of sensor resources to the application requirements in terms of latency, data rate and lifetime [19].

In this paper we demonstrate how the performance of a WSN network can be further improved, if nodes not only synchronize, but also *desynchronize* with one another. Desynchronization refers to the term where nodes on different branches of the routing tree are active at different times to avoid radio interference. In WSNs nodes can be logically organized in groups. More precisely, the activity schedules of nodes that need to communicate with one another are synchronized to improve message throughput. We say that those nodes belong to one *coalition*. At the same time, the schedules of groups of nodes which do not need to communicate are desynchronized in order to avoid radio interferences and packet losses. We refer to this type of coordination for short as *(de)synchronization*.

We show that coordinating the activities of the sensor nodes can successfully be done using a win-stay lose-shift (WSLS) strategy, drawn from game theory. We call the approach DESYDE, which stands for DEcentralized SYNchronization and DESynchronization. The coordination is achieved by rewarding successful interactions (e.g., transmission of a message) and penalizing the ones with a negative outcome (e.g., message loss or overhearing). This behavior drives the sensor nodes to repeat actions that result in positive feedback more often and to decrease the probability of unsuccessful interactions. Nodes that tend to select the same successful action naturally form a coalition. The main benefit of the proposed approach is that global (de)synchronization emerges from simple and local interactions without the need of central mediator or any form of explicit coordination. An additional advantage is that DESYDE works with any routing algorithm that forms a routing tree connecting the nodes to the base station.

We implement DESYDE in the OMNeT++ simulator [9], and study three different wireless sensor network topologies, namely line, grid, and random. We compare it to S-MAC [20] and D-MAC [12], two state-of-the-art coordination mechanisms for WSNs, and show that nodes form coalitions which improve data communication and reduce packet collisions. This enables a quicker delivery of the data packets to the base station, allowing shorter active periods and lower energy consumption.

The rest of the paper is organized as follows: Section 2 presents the background of our research. It outlines the application domain, explains the communication and routing protocols and guides the reader through related work. The DESYDE approach is described in Section 3, and experimentally compared on different topologies in Section 4. We finally discuss the results in Section 5 shortly before we conclude in Section 6.

2. BACKGROUND AND RELATED WORK

A Wireless Sensor Network is a collection of densely deployed autonomous devices, called *sensor nodes*, which gather data with the help of sensors [10, 2]. The untethered nodes use radio communication to transmit sensor measurements to a terminal node, called the base station or *sink*. The sink is the access point of the observer, who is able to process the distributed measurements and obtain useful information about the monitored environment. Sensor nodes communicate over a wireless medium, by using a multi-hop communication protocol that allows data packets to be forwarded

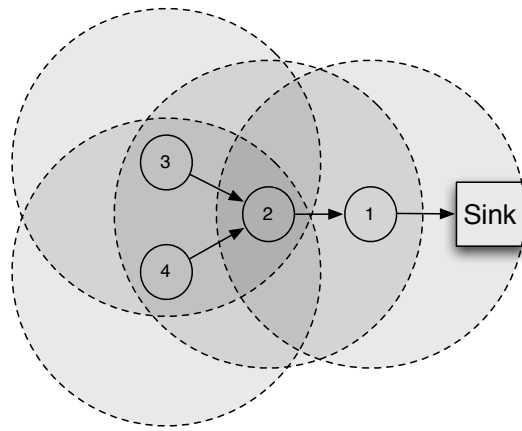


Figure 1: Sensor nodes connected to a base station by means of a multi-hop routing tree. Grayed circles indicate overlapping communication regions.

by neighboring nodes to the sink.


When the WSN is deployed, the routing protocol requires that the nodes determine a routing path to the sink [3, 10]. This is achieved by letting nodes broadcast packets immediately after deployment in order to discover their neighbors. Nodes in communication range of the sink propagate this information to the rest of the network. During the propagation process, each node chooses a *parent*, i.e. a node to which the data will be forwarded in order to reach the sink. The choice of a parent can be done using different metrics, the standard one being the hop distance, i.e. the minimum number of nodes that will have to forward their packets [4, 18]. An example of multi-hop shortest path routing structure is given in Fig. 1, together with the radio communication ranges of sensor nodes.

Since wireless sensor nodes operate in most cases on finite energy resource, low-power operation is one of the crucial design requirements in sensor networks [2, 10]. The challenge of energy-efficient operation must be tackled on all levels of the network stack, from hardware devices to protocols and applications. Although sensing and data processing may incur significant energy consumption, it is commonly admitted that most of the energy consumption is caused by the radio communication. A large amount of research has therefore been devoted in recent years to the design of energy-efficient communication protocols [10, 20].

Fig. 2 reports the radio characteristics of two representative and often used radio platforms: the CC2420 (used in TelosB and IMote2) and the Xbee-802-15.4 (used in the Waspote). An important observation is that for these typical radios, the sleep power is at least two orders of magnitude lower than the transmit and receive power. Therefore, the only way to significantly reduce power consumption is to have the radio switched off most of the time, and to turn it on only if messages must be received or sent. This problem is referred to as *wake-up scheduling*.

Wake-up scheduling in wireless sensor networks is an active research domain, and a good survey on wake-up strategies in WSNs is presented in [19]. Three types of wake-up solutions can be identified, namely, on-demand paging, synchronous and asynchronous wake-up.

In on-demand paging, the wake-up functionality is man-



Mote Year	Tmote Sky 2005	Imote 2 2007	Waspote 2009
Radio	CC2420		Xbee-802.15.4
Outdoor range	50/100m		500 m
Data rate	250 Kbps		20 Kbps
Sleep power	60 μ W sleep		<30 μ W sleep
Receiving power	63 mW receive		150 mW receive
Transmit power	57 mW wmit		135 mW xmit
Startup time	1 ms setup		2 ms setup

Figure 2: Typical wireless sensor hardware developed in the recent years, together with their main radio characteristics.

aged by a separate radio device, which consumes much less power in the idle state than the main radio. The main radio therefore remains in a sleeping state, until the secondary radio device signals that a message is to be received on the radio channel. This idea was first proposed with the PicoRadio and PicoNode projects [7] for extremely low power systems, and extended in [16, 1] with hand-held devices. On-demand paging is the most flexible and energy-efficient solution, but adds non-negligible costs in the hardware design.

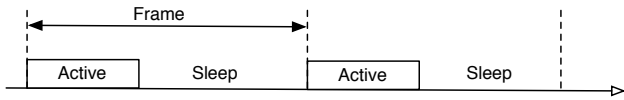


Figure 3: Structure of S-MAC with duty cycle and synchronous wake-up scheduling.

In synchronous wake-up approaches, nodes duty-cycle their radio in a coordinated fashion. Several MAC (Medium Access Control) protocols have been proposed, allowing nodes to wake-up at predetermined periods in time at which communication between nodes becomes possible. A standard paper detailing this idea is that of S-MAC (Sensor-MAC) [20]. The basic scheme is that nodes rely on a fixed duty-cycle, specified by the user, where nodes periodically switch between the active and sleep states. The period is called a frame, and an example of periodic schedule is illustrated in Fig. 3. Several extensions to S-MAC have been proposed. In particular, authors in [12] proposed D-MAC, which aims at improving the efficiency by both reducing the latency and the active period of the sensor nodes. This is achieved by dividing the frame into slots, and by staggering the wake-up cycles along the routing tree, as illustrated in Fig. 4. The active period consists in receiving and sending only one data packet, and active periods are staggered so that nodes send data when their parent's radio is in the receive mode. If multiple packets need to be sent by a node, a *more data* flag is set to warn the parent node that additional data must still be received. If the flag is set, the node and its parent schedule the pending communication to happen after a small backoff period (a few milliseconds), to avoid collisions. This approach was recognized in the MAC review of Langendoen [11] to be particularly compelling for collecting data from a WSN in a timely and energy-efficient manner. The main concern with protocols based on synchronous wake-up is however the overhead which can be caused by maintaining

the nodes synchronized.

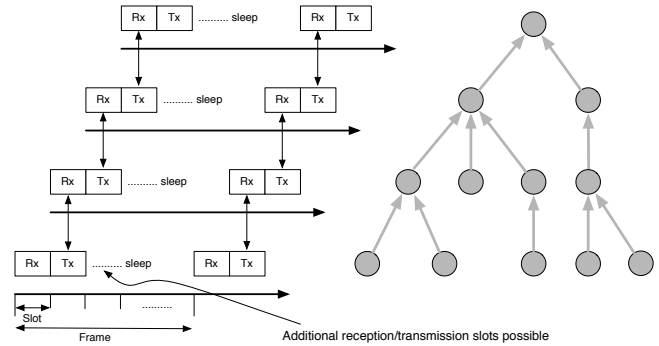


Figure 4: D-MAC scheduling protocol: Nodes are synchronized so that data flows from leaf nodes to the root node in the routing tree.

Finally, in asynchronous wake-up solutions, the wake-up schedules need not be coordinated, and may possibly be different. The communication therefore comes at an increase cost for either the sender or the receiver. In sender-based asynchronous wake-up, the sender continuously sends beacons until the receiver is awake. Once the receiver gets the beacon, it sends an acknowledgment to notify the sender that it is ready to receive a packet. This scheme is the basis for the low-power listening [8] and preamble sampling [5] protocols. The receiver-based wake-up solution is the mirror image of sender-based, and was exposed in the Etiquette protocol [6]. Sender-based and receiver-based asynchronous protocols can achieve very low power consumption. Asynchronous wake-up solutions however require an overhead due to the signaling of wake-up events, which makes them inefficient when wake-up events are relatively frequent [19].

3. DESYDE

This section presents DESYDE, an acronym for DEcentralized SYNchronization and DESynchronization, which aims at improving communication performances in wireless sensor networks by coordinating the radio activity of neighboring sensor nodes. Our approach belongs to the category of synchronous wake-up strategies, and is intuitively motivated by an important (although not apparent at first sight) weakness of D-MAC. Recall from the previous section that D-MAC schedules the radio activity of sensor nodes in such a way that children and parents in the routing tree synchronize their radio transmission/reception slots. While this strategy appears at first sight to offer great benefits over S-MAC, it only works well if nodes are arranged in a line topology (cf. Fig. 5, middle). Indeed, whenever the routing tree contains several branches, neighboring nodes which are the same number of hops away from the base station may interfere, causing packet losses, and possibly important delays (once a transmission fails in D-MAC, the packet is queued until the next frame).

A better schedule is therefore one where nodes along the same branch of a routing tree are staggered, as in D-MAC (so that end-to-end latency is improved), while at the same time being desynchronized with nodes on neighboring branches of the routing tree (so that communication interference is minimized). Synchronized groups of nodes will be referred

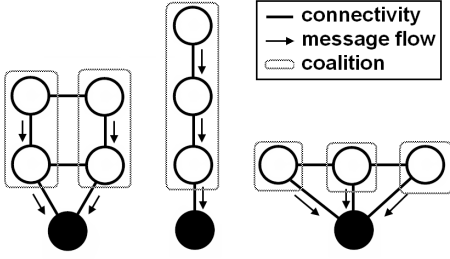


Figure 5: Examples of routing and coalition formation

to as *coalitions*. Figure 5 illustrates the concept of coalitions in three different topologies. Intuitively, nodes on the same branch of a routing tree should form a coalition so that data can be relayed efficiently from a leaf node to the root node. At the same time, nodes which are the same same number of hops away from the base station should desynchronize since they belong to separate branches of the routing tree.

The resulting schedule of DESYDE for the 2 by 2 grid in Fig. 5 (left) is illustrated in Fig. 6. In this example, the frame contains 10 slots, and the four schedules reported are those of the four nodes in the grid, arranged in the same order as in Fig. 5 (left). At slot 2, the upper left node transmits when the lower left node receives, while the right nodes are synchronized for communication at slot 5. The lower left node send its data to the base station at slot 7 and forwards that of the upper left node at slot 9. The lower right node does the same at slots 4 and 6, respectively. Thus, we observe the same coalitions as in our schematic model in Figure 5 (left).

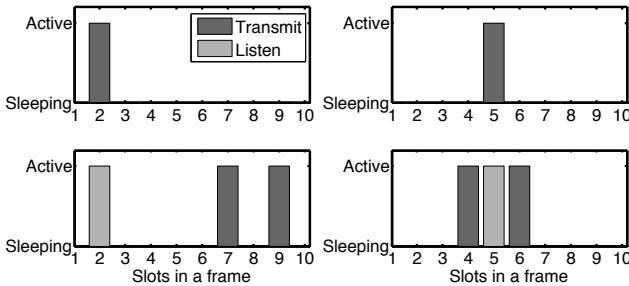


Figure 6: Examples of DESYDE schedule for the 2x2 grid. Upper transmission slots are synchronized with lower reception slots. Left active slots are desynchronized with right active slots.

3.1 Learning Model

We present here the underlying approach that makes nodes coordinate their behavior in a decentralized manner. We see the WSN as a Multi-Agent System (MAS), where agents are the sensor nodes. Recall that the frame stands for the fixed-length activity period that nodes periodically repeat over time. Each frame is divided in a number of slots that constitute discrete time units, where agents can select only one of three actions, namely *transmit*, *listen* or *sleep*.

At each slot, agents are therefore involved in a game with their neighbors where all nodes have to independently choose

an action. The goal of agents is to forward all their messages to the sink, minimizing the end-to-end latency of packets. Upon executing any of the three actions, each agent receives feedback from the environment. This feedback depends on the *event* that the actions produced (for example successful transmission, collision, or idle listening). These events will be detailed in section 3.2, and may be either successful or unsuccessful. The success of the resulting event is used by the node to assess the quality of its action.

One can notice that the game our agents are involved in at each time slot has certain characteristics of Graphical Games [17], where neighboring agents have to independently decide on an action. In our network, however, the graphical game at each time step is dependent on the one played at the previous slot, due to forwarding of messages. In other words, our agents are engaged in sequential graphical games, where each game is closely related to the preceding one. The dependence between these sequential games is a result of the forwarding of packets between nodes. Put differently, the transmission of messages between neighbors influences their choice of action at each time step. These actions, however, have no immediate effect on agents further in the network.

Formally, we represent our learning system with the following notation:

- Each frame F is divided in N slots, and the set of time slots in a frame is denoted $S = \{s^1, \dots, s^N\}$.
- A is the set of available actions a for each agent at each slot.
- $R : S \times A \rightarrow [0, 1]$ is the reward signal $R(s, a)$ for taking action a in slot s .

As specified above, the action space A at each time slot s for each agent is identical and restricted to the following three actions: $a_{transmit}$, a_{listen} and a_{sleep} . At every time step each agent may detect a communication event, caused by its own actions and those of its neighbors. Upon executing action a , the agent receives a reward $R(s, a)$, determined by the outcome of the actions its neighbors chose at slot s and its own action. This interaction between neighboring nodes is further elaborated in the following subsections.

3.2 Rewards, Updates and Action Selection

We use $Q(s, a)$ to indicate the expected reward (or “quality”) of taking action a at slot s . It represents the latest reward obtained at that slot for that action. At first, this value is initialized to 0 for $Q(s, a_{transmit})$ and $Q(s, a_{sleep})$ and 1 for $Q(s, a_{listen})$. Upon executing action a at slot s , the agent updates its action quality, based on the reward it receives: $Q(s, a) \leftarrow R(s, a)$. Note that only one of the three actions can be taken during a slot. Therefore, at every slot s , $Q(s, a)$ is 1 for exactly one action a and 0 for the two others.

We classify each communication event, that a node can detect, using a boolean value to signalize whether the event was positive or negative for that node. Based on our simple update rule a boolean representation is sufficient in our learning model. We modeled six different events, namely successful transmission (if ACK received), successful reception, overhearing, idle listening, unsuccessful transmission (if no ACK received), and collision. We consider these six events to be the most energy expensive or latency crucial in wireless communication. The reward for the two events is 1

(successful transmission or reception) and for the rest it is 0.

Recall that only one action can have a quality of 1 for each slot. We use policy $\pi(s)$ to denote the action a that the agent selects at slot s , based on the quality $Q(s, a)$ for that action in that slot. More precisely,

$$\pi(s) = \begin{cases} a_{\text{transmit}}, & \text{if } Q(s, a_{\text{transmit}}) = 1 \\ a_{\text{listen}}, & \text{if } Q(s, a_{\text{listen}}) = 1 \\ a_{\text{sleep}}, & \text{if } Q(s, a_{\text{sleep}}) = 1 \end{cases} \quad (1)$$

This behavior resembles a *win-stay lose-shift strategy* [15], where agents repeat successful actions and avoid unsuccessful ones. In particular, at slot s an agent will repeat action a only if it had a positive outcome at slot s in the previous frame. Recall that frames capture the periodic behavior of nodes. Thus, in every frame the agent repeats those actions that had positive outcome in the previous frame. For example, according to policy $\pi(s)$, if $Q(s, a_{\text{transmit}}) = 1$ for slot s , the node will choose to transmit a packet during that slot (provided that it has a packet in its queue). As a result, a reward $R(s, a_{\text{transmit}})$ will be generated and stored in $Q(s, a_{\text{transmit}})$. If its transmission was acknowledged, $Q(s, a_{\text{transmit}})$ will stay 1 and the agent will repeat the same action next frame at slot s . Otherwise, based on the event that occurred, it will choose a different action. In the same way the agent will select an action in every slot within the frame F .

3.3 Exploration

We would like to note that the outcome of each of the actions a_{transmit} and a_{listen} gives us information about the quality of the other two actions as well. Both these actions require that the radio transmitter of the node is switched on, which enables the agent to detect events in its environment and therefore obtain feedback. Consider the following example. If $\pi(s) = a_{\text{transmit}}$, upon observing the outcome the agent will know whether it would have been more beneficial to *listen* or *sleep* during slot s instead. Provided that its message is acknowledged, transmitting is indeed the best action. If no acknowledgment is received, its parent is probably busy (or sleeping) and therefore it would be better to *listen* for packets at slot s next frame, rather than *transmit*. In case it detects a collision during that slot, for next frame *sleeping* would be most beneficial in order to avoid receiving in vain. The same reasoning holds if the agent selects action a_{listen} .

Action a_{sleep} on the other hand turns the energy-consuming antenna off and consequently prevents the agent from receiving any information from its environment. For this reason, during a short exploration stage, fixed by the user, the agent never selects the latter action to keep its radio transmitter on. In other words, during exploration if $\pi(s) = a_{\text{sleep}}$ the agent will select a_{listen} at that slot instead. This behavior enables the node to constantly acquire feedback and therefore update its quality values. In the WSN domain, such an exploration is very costly in terms of battery consumption. However, we determined empirically that nodes require no more than 3 to 4 frames of exploration for their policies to converge. After the exploration stage expires, the agent reverts to the policy described in Formula 1. Intuitively, when an agent finds a “win” action for a certain slot, its *win-stay lose-shift* strategy will prevent it from choosing a different action at the same slot next frame. Thus, every

agent learns a periodical schedule based on the events that happen as a result of its actions. We therefore say that no explicit form of agent coordination is necessary to achieve equilibrium. Instead, coordination “emerges” as a result of packet forwarding and reasoning based on local interactions.

4. EXPERIMENTAL STUDY

4.1 Experimental Setup

We apply our approach on three networks of different size and topology – a 4-hop line, a 16-node (4 by 4) grid topology and one with 50 nodes scattered randomly with an average of 5 neighbors per node. The first topology requires nodes to synchronize in order to successfully forward messages to the sink. Intuitively, if any one node is awake while the others are asleep, that node would not be able to forward its messages to the sink. The second topology illustrates the importance of combining synchronization and desynchronization, as neither one of the two behaviors alone is an efficient strategy. The random topology shows the scalability of our approach to larger networks where the topology is not known a priori. In our simulations we use a shortest path routing scheme that creates a static routing tree. A similar experimental setup is also used in [13, 14].

Each of the three networks was ran for 200 seconds in the OMNeT++ simulator [9] and results were averaged over 30 runs. This network runtime was sufficiently long to eliminate any initial transient effects. To illustrate the performance of the network at high data rates, we set the sampling period of nodes to one message every 10 seconds. To simulate periodic data collection, this message is generated at the beginning of each frame for all nodes. Frames have the same length as the sampling period and were divided in $N = 2000$ slots of 5 milliseconds each. The duration of the slot was chosen such that only one DATA packet can be sent and acknowledged within that time. All hardware-specific parameters, such as transmission power, bit rate, etc., were set according to the data sheet of our radio chip — CC2420 (cf. Table 2). In addition, we chose the protocol-specific parameters, such as packet header length and number of retransmission retries as specified in the IEEE 802.15.4 communication protocol. We set the duration of the exploration stage to 5 frames, or 50 seconds. It was empirically measured that this duration was enough for the policies of all agents to converge.

To better illustrate the importance and effect of combining synchronization and desynchronization in these topologies, we compare our approach to two state-of-the-art MAC protocols, viz. S-MAC and D-MAC. In addition, we present the case where all nodes remain active for the entire duration of the simulation and never switch off their radio transmitter (called ALL-ON for short). The latter behavior serves only as a benchmark in terms of end-to-end latency, because the energy consumption of this protocol renders it impractical for real-world scenarios. Since nodes have a duty cycle of 100%, packets will not experience any sleep latency and will be quickly forwarded to the sink. The S-MAC protocol illustrates network performance under synchronized behavior, where all nodes are active at the same time. D-MAC on the other hand shows whether staggering the short duty cycles across hops is an efficient strategy to improve latency and lifetime.

4.2 Evaluation

Figures 7 and 8 display the average battery consumption and latency for all three topology using the S-MAC protocol. According to S-MAC, all nodes wake up at the beginning of the frame for a duration specified by the user. We therefore vary the duty cycle and observe the performance of the system in terms of lifetime and throughput. Intuitively, the energy consumption under S-MAC increases linearly with increasing duty cycle for all three topologies, as Figure 7 displays.

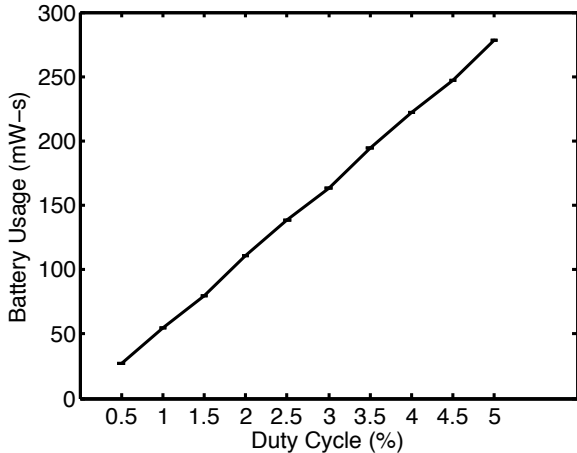


Figure 7: Average battery usage under the S-MAC protocol for different duty cycles (same for all topologies)

Since collisions constitute the biggest obstacle in the pursuit of low latency, each node contends for the channel for a small random duration within a fixed contention window. To facilitate the throughput of messages at high data rates, we deviated from the contention policy of S-MAC that uses the entire active time as a contention window. Instead, in our simulations we fixed the maximum contention window of S-MAC to 5 slots for a more fair comparison. Even though short duty cycles are appealing from an energy perspective, nodes do not manage to forward all their packets within one active interval for all three topologies. The latter result can be observed in Figure 8, where error bars signify one standard deviation across 30 runs. For duty cycles below 2% (or 1% in the line topology) nodes do not manage to forward all their packets within one active interval. The reason for this high latency is the large number of collisions when all nodes wake up at the same time. This phenomenon is particularly visible in the grid topology, where every node has exactly one parent and at least one neighbor who belongs to a different branch of the routing tree.

For duty cycles larger than 2% nodes in the line and grid topologies manage to send all their packets within one active interval and therefore the average end-to-end latency is reduced to around 0.1 seconds. Nodes in the random network, however, require two active periods to forward their messages and therefore the latency settles at around 10 seconds, or 1 frame duration. Still, we measured around 20% packet loss on average for the latter topology, due to the large number of retransmissions necessary when all nodes

are active at the same time. The large standard deviation is due to the different random topologies across runs.

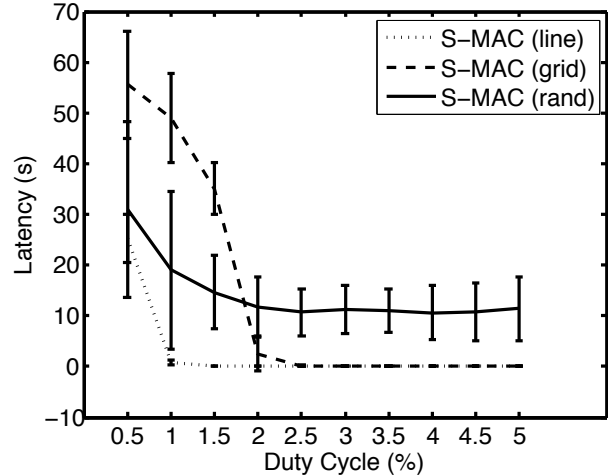


Figure 8: End-to-end latency under the S-MAC protocol for different duty cycles

S-MAC illustrates the effect of full synchronization on the network performance. We see that large networks suffer from high sleep latency for small duty cycles. D-MAC on the other hand makes sure that sleep latency is reduced by “staggering” the wake-up cycles of nodes according to their hop distance to the sink (cf. Figure 4). In other words, all nodes that lie at the same distance from the sink are synchronized to wake up at the same time and send a packet to their parents, who wake up at the slot just after their children. This behavior is designed to reduce the sleep latency that S-MAC suffers from. The duty cycle of each node under D-MAC is dependent on its traffic load (i.e., its position in the data gathering tree), as it is the case with our learning algorithm. Similar to S-MAC, to reduce collisions we let each node contend for the channel for a small random time within a fixed contention window. The size of this window, however, affects performance. Large contention window will lower the probability of collision, but at the cost of delayed transmissions. A small one, on the other hand, will speed up communication, but will cause more unsuccessful transmissions. We therefore present the performance of each protocol for different contention window sizes. To abide by the specifications of D-MAC, we define the size of its contention window in terms of the duration of a DATA packet. The design of DESYDE, however, requires us to set the contention window as a factor of the slot length instead (which is a DATA packet + an acknowledgment). We use the latter setting in S-MAC and ALL-ON as well. Since the difference between the two contention windows is negligible, we use the same axis in Figures 9 and 10 to plot the performance of both protocols. We write “contention window size” to signify the *factor* of the contention window, which is fixed for each run. Due to space limitations we graph the battery consumption of D-MAC and DESYDE for each topology on the same plot.

In Figure 9 we see that in all three topologies DESYDE outperforms D-MAC in terms of energy consumption, irrespective of the contention window size. Due to the “win-stay

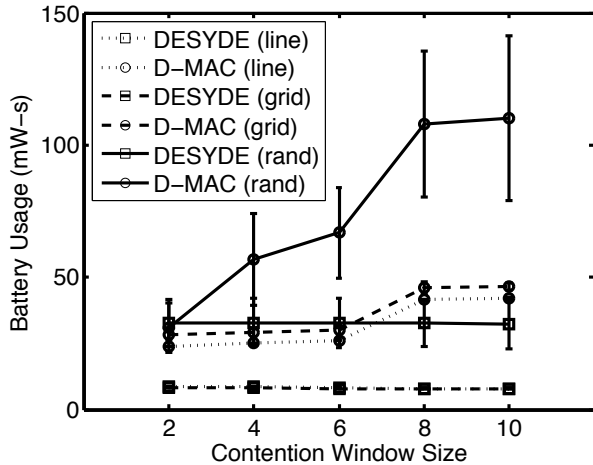


Figure 9: Battery usage under D-MAC and DESYDE for different contention windows across different topologies

lose-shift” strategy, after the exploration stage our learning algorithm remains invariant to the contention window size. In other words, contention is used only during exploration. Each node, thereafter, learns to transmit in a different time slot within the frame and thus contention for the channel is not necessary. Nodes under D-MAC, however, always wake up for one listen and one transmit slot, regardless of the node position in the network. A disadvantage is that leaf nodes still listen for one slot, when they need not, while all other nodes need to hold an additional listen + transmit slot for every packet they generate. The energy consumption of D-MAC is therefore higher than the one of DESYDE for each topology. Moreover, according to specifications the active period of D-MAC includes the time for channel contention. Therefore its battery consumption increases with the size of the contention window.

Lastly, we present the difference between ALL-ON, DESYDE and D-MAC in terms of the end-to-end latency averaged over 30 random topologies, each consisting of 50 nodes. Figure 10 compares the three protocols for different contention windows. One can notice that DESYDE once again outperforms D-MAC. DESYDE enables nodes to both synchronize with their parents and desynchronize with their same-hop neighbors. Recall the example we presented in Figure 6, which shows the resulting schedule of nodes under DESYDE in a sample 2 by 2 grid topology. In some routing schemes, such as ours, all nodes that lie on the same hop belong to different branches of the routing tree. In D-MAC, however, all those nodes wake up at the same time and therefore cause radio interferences, followed by packet retransmissions. Intuitively, latency under D-MAC decreases for larger contention windows, but nodes still require more than one active period to deliver all their packets. DESYDE, on the other hand, has comparable latency to ALL-ON, where nodes never switch off their antenna and therefore packets incur no sleep delay. While ALL-ON requires 100% duty cycle, DESYDE is able to achieve the same latency with only 0.8% active time within a frame.

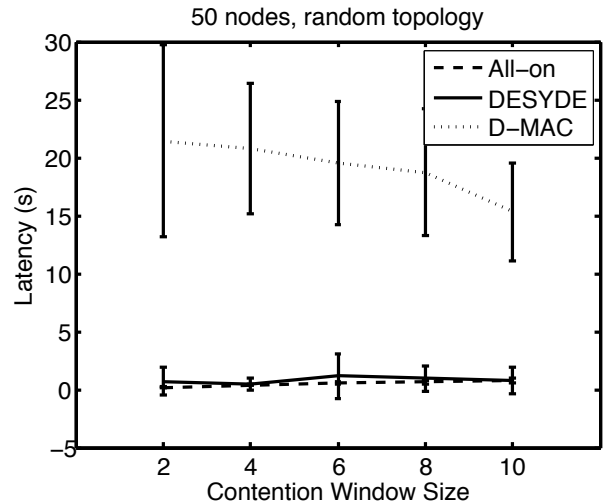


Figure 10: End-to-end latency under ALL-ON, DESYDE and D-MAC for different contention windows

5. DISCUSSION AND FUTURE WORK

The experimental results presented in the previous section illustrate that DESYDE is able to significantly improve the performance of a data collection task in wireless sensor networks. The two main metrics considered were the latency and the energy consumption. For both metrics, large gains could be observed, over a wide range of networking parameters. These results were particularly remarkable for large and random topologies. The main reason is that DESYDE relies on a learning strategy which can adapt to complex topologies and traffic patterns.

The *win-stay lose-shift* (WSLS) strategy which underlies DESYDE is a key aspect of the proposed approach. Several research directions can be pursued in order to further improve its performance. First, an advantage of the WSLS is that it provides a way to reduce the exploration space and to accelerate the convergence of the learning stage. A direct drawback of this “aggressive” exploration is that more efficient solutions to the coordination of sensor nodes may be too quickly discarded. One of the research axes we plan to focus on consists in relying on “smoother” updating rules for the quality values of the actions. This could be done by using a learning factor which keeps tracks of past rewards during the learning process.

A second important parameter is the convergence time of the learning process. We observed in all our experiments that this time is in practice very short, in the order of a few data collection rounds (around 5). It is still unclear under which conditions convergence proofs can be brought. One can easily notice that, using the WSLS approach, the convergence is guaranteed if no node modifies its policy for two consecutive rounds. This unfortunately does not seem to be detectable without all nodes exchanging information about their status, which would be energy costly. Further research is therefore required to better characterize the convergence criteria.

Finally, we assumed, as most protocols which fall in the synchronous category, that the traffic patterns and the net-

work topology are stationary for every run. As proposed, DESYDE is not robust to topology changes, or to variations in the data collection rate. The common solutions to these issues is to rely on periodic checks concerning the amount of dropped packets, or queue sizes on the sensor nodes, and to restart the coordination of the nodes if necessary. While not specific to the approach presented in this paper, further research is also required in this area.

6. CONCLUSION

Synchronous wake-up strategies can greatly reduce the duty cycle of sensor nodes in a WSN. We however highlighted in this paper that they suffer from potential high latency and energy waste due to radio interferences and packet collisions. These deficiencies stems from the fact that neighboring sensor nodes should only synchronize their activities when they engage in a communication, and *desynchronize* otherwise. The proposed approach, DESYDE, a DEcentralized SYNchronization DESynchronization strategy, aims at tackling this problem. The core of the approach is based on a win-stay lose-switch strategy, which we implement in a fully decentralized way.

Our OMNeT++ implementation showed that state-of-the-art synchronized protocols only perform well in simple networks, such as line topologies. As the network complexity grows, these protocols resulted in high latency and energy costs, due to the increased number of packet collisions and packet retransmissions. DESYDE was able in all our experiments to compete with standard approaches, and exhibited significant gains in latency and energy especially for larger networks.

Acknowledgments

The authors of this paper would like to thank the anonymous reviewers for their useful comments and valuable suggestions. This research is funded by the agency for Innovation by Science and Technology (IWT), project DiCoMAS (IWT60837); and by the Research Foundation – Flanders (FWO), project G.0219.09N.

7. REFERENCES

- [1] Y. Agarwal, R. Gupta, and C. Schurgers. Dynamic power management using on demand paging for networked embedded systems. In *Proceedings of the Asia and South Pacific Design Automation Conference*, volume 2, pages 755–759, 2005.
- [2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks. *Communications Magazine, IEEE*, 40(8):102–114, 2002.
- [3] J. Al-Karaki and A. Kamal. Routing techniques in wireless sensor networks: a survey. *Wireless Communications, IEEE*, 11(6):6–28, 2004.
- [4] D. Couto, D. Aguayo, J. Bicket, and R. Morris. A high-throughput path metric for multi-hop wireless routing. *Wireless Networks*, 11(4):419–434, 2005.
- [5] A. El-Hoiydi. Aloha with preamble sampling for sporadic traffic in ad hoc wireless sensor networks. In *IEEE International Conference on Communications*, volume 5, pages 3418–3423, 2002.
- [6] S. Goel. *Etiquette protocol for ultra low power operation in energy constrained sensor networks*. PhD thesis, New Brunswick, NJ, USA, 2005.
- [7] C. Guo, L. Zhong, and J. Rabaey. Low power distributed MAC for ad hoc sensor radio networks. *GLOBECOM*, 5:2944–2948, 2001.
- [8] J. Hill and D. Culler. Mica: A wireless platform for deeply embedded networks. *IEEE micro*, 22(6):12–24, 2002.
- [9] <http://www.omnetpp.org/> – a C++ simulation library and framework.
- [10] M. Ilyas and I. Mahgoub. *Handbook of sensor networks: compact wireless and wired sensing systems*. CRC, 2005.
- [11] K. Langendoen. Medium access control in wireless sensor networks. *Medium access control in wireless networks*, 2:535–560, 2008.
- [12] G. Lu, B. Krishnamachari, and C. Raghavendra. An adaptive energy-efficient and low-latency MAC for data gathering in wireless sensor networks. In *Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International*, page 224, 2004.
- [13] M. Mihaylov, Y.-A. Le Borgne, K. Tuyls, and A. Nowé. Decentralised Reinforcement Learning for Energy-Efficient Scheduling in Wireless Sensor Networks. *International Journal of Communication Networks and Distributed Systems*, 6, 2011. To appear.
- [14] M. Mihaylov, Y.-A. Le Borgne, K. Tuyls, and A. Nowé. Self-Organizing Synchronicity and Desynchronicity using Reinforcement Learning. In *Proceedings of the 3rd International Conference on Agents and Artificial Intelligence*, pages 94–103, Rome, Italy, 2011.
- [15] M. Posch. Win-Stay, Lose-Shift Strategies for Repeated Games–Memory Length, Aspiration Levels and Noise. *Journal of theoretical biology*, 198(2):183–195, 1999.
- [16] E. Shih, P. Bahl, and M. Sinclair. Wake on wireless: An event driven energy saving strategy for battery operated devices. In *Proceedings of the 8th annual international conference on Mobile computing and networking*, pages 160–171, 2002.
- [17] D. Vickrey and D. Koller. Multi-agent algorithms for solving graphical games. In *Proceedings of the National Conference on Artificial Intelligence*, pages 345–351. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2002.
- [18] A. Woo, T. Tong, and D. Culler. Taming the underlying challenges of reliable multihop routing in sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 14–27. ACM, 2003.
- [19] W. W. Y. Li, M.T. Thai, editor. *Wireless Sensor Networks and Applications*, chapter Wakeup Strategies in Wireless Sensor Networks, page 195. Springer, 2008.
- [20] W. Ye, J. Heidemann, and D. Estrin. Medium access control with coordinated adaptive sleeping for wireless sensor networks. *IEEE/ACM Trans. Netw.*, 12(3):493–506, 2004.

Logic-Based Approaches I

A Framework for Coalitional Normative Systems

Jun Wu^{1,2}, Chongjun Wang^{1*} and Junyuan Xie¹

¹National Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210093, China

²College of Computer & Information Engineering, Hohai University, Nanjing 210098, China
wujun@iip.nju.edu.cn, {chjwang, jyxie}@nju.edu.cn

ABSTRACT

We propose coalitional normative system (CNS), which can selectively restrict the joint behavior of a coalition, in this paper. We extend the semantics of ATL and propose *Coordinated* ATL (CO-ATL) to support the formalizing of CNS. We soundly and completely characterize the limitation of the normative power of a coalition by identifying two fragments of CO-ATL language corresponding to two types of system properties that are unchangeable by restricting the joint behavior of such a coalition. Then, we prove that the effectiveness checking, feasibility and synthesis problems of CNS are PTIME-complete, NP-complete and FNP-complete, respectively. Moreover, we define two concepts of optimality for CNS, that is, minimality and compactness, and prove that both minimality checking and compactness checking are co-NP-complete while the problem of checking whether a coalition is a minimal controllable coalition is DP-complete. The relation between NS and CNS is discussed, and it turns out that NSS intrinsically consists of a proper subset of CNSS and some basic problems related to CNS are no more complex than that of NS.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent systems; I.2.4 [Knowledge representation formalisms and methods]

General Terms

Theory

Keywords

normative systems, logic, model checking, complexity

1. INTRODUCTION

Normative system (NS) (or *social law*) was firstly proposed by [12, 13] as an off-line approach for coordinating multiagent systems, and then extended by, e.g., [17, 14, 1], based on introducing the formalisms of modal and temporal logics,

*Chongjun Wang is the corresponding author.

Cite as: A Framework for Coalitional Normative Systems, Jun Wu, Chongjun Wang and Junyuan Xie, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 259-266.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

especially *Alternating-time Temporal Logic* (ATL) [4, 5] and its variations, which can be used for the specification and verification of mechanisms such as social choice procedures. So work on this aspect is also considered as a part of the *logics for automated mechanism design* research [10, 16].

Although the various approaches to NSS proposed in the literature differ on technical details, they all share the same basic intuition that an NS is a set of constraints on the behavior of agents; by imposing these constraints, it is hoped that some desirable objectives will emerge [3], corresponding to a logic formula that is originally false to become true, or the reverse. The idea is that the imposing of an NS will lead to certain updating in the semantic model, and thus cause changes in the interpretations of some formulas.

But we find that NSS update the semantic model in a somewhat too coarse way, that is, when an action is forbidden in a state, all related transitions from this state are deleted¹. This means the task of deleting a certain set of prescribed transitions, which corresponds to the necessary condition for fulfilling a certain objective, may exceed the abilities of all NSS. To overcome this shortcoming, we propose coalitional normative system (CNS), which is a set of behavioral constraints for a coalition (*i.e.*, agent set) that restrict its *joint actions*. By adopting a CNS, we can restrict the set of transitions to an arbitrary subset of it, thus we can achieve all possible updating in the semantic model.

Intuitively, the coalition represents a system we can control (it is a distributed open system formed by several agents); and the CNS specifies in every state for the coalition which sets of actions (that can be chosen by it) cannot be executed simultaneously, thus should be forbidden. We assume that the agents in the coalition will negotiate with each other before making any decisions on action selection in order to avoid adopting any joint actions that are forbidden by the CNS. So, compared with conventional NS, CNS can more effectively capture the overall effects of joint actions and prevent the destructive interactions from taking place. In this paper, we aim to present a framework for CNS based on ATL, and study its related reasoning and computational problems.

The remainder of this paper is structured as follows. We begin by introducing the basics of ATL and NS. Next, as the effects of CNSS cannot be captured by ATL directly, we propose CO-ATL to support the formalizing of CNS. Then for each coalition C , by identifying the \mathcal{L}_C^+ and \mathcal{L}_C^- fragments of the CO-ATL language, we “*soundly and completely*” characterize its limitation of normative power. Afterward, we

¹When concurrent action models are adopted, an action of an agent in a state may be related to several transitions.

establish the computational complexity of some key problems related to CNSS. Finally, we present some conclusions.

2. ATL AND NORMATIVE SYSTEMS

2.1 Alternating-time Temporal Logic

The syntax of *Alternating-time Temporal Logic* (ATL) [4, 5] is an extension of the syntax of CTL via replacing the path quantifiers \forall and \exists with the path quantifier $\langle\langle\rangle\rangle$, which can express the α -ability [15] in game theory. And the formulas of ATL are interpreted by *concurrent game structures* (CGSS).

A concurrent game structure is a tuple $S = \langle k, Q, \Pi, \pi, d, \delta \rangle$ with the following components:

- A natural number $k \geq 1$ of agents. We identify the agents with the numbers (or IDs) $1, \dots, k$.
- A finite set Q of states.
- A finite set Π of propositions.
- For each state $q \in Q$, a set $\pi(q) \subseteq \Pi$ of propositions true at q . The function π is called labeling function.
- For each agent $a \in \{1, \dots, k\}$ and each state $q \in Q$, a natural number $d_a(q) \geq 1$ of actions available to agent a at state q . We identify the actions of agent a at state q with the numbers $1, \dots, d_a(q)$. A *joint action* of all the agents at state q is a tuple $\langle j_1, \dots, j_k \rangle$ such that $1 \leq j_a \leq d_a(q)$ for each agent a . We write $D(q)$ for the set $\{1, \dots, d_1(q)\} \times \dots \times \{1, \dots, d_k(q)\}$ of joint actions. The function D is called move function.
- For each state $q \in Q$ and each joint action $\langle j_1, \dots, j_k \rangle \in D(q)$, a state $\delta(q, j_1, \dots, j_k) \in Q$ will result from state q if every agent $a \in \{1, \dots, k\}$ chooses action j_a . The function δ is called transition function.

Note that, every agent set $A \subseteq \{1, \dots, k\}$ can be seen as a coalition (with the agent set $\{1, \dots, k\} \setminus A$ represents the environment). The *grand coalition* $\{1, \dots, k\}$ is denoted as Ag . In the following of this paper, we will sometimes use \vec{m} to refer to a joint action $\langle j_1, \dots, j_k \rangle$ (of all the agents), use \vec{m}_A to refer to a joint action of the coalition $A \subset Ag^2$ (called an *A-action*) and use $D_A(q)$ to refer to all the possible A-actions at the state q . Moreover, we introduce the notation $\vec{m}_A|A'$ to mean the joint action of the agent set $A \cap A'$ when the agent set A takes the joint action \vec{m}_A .

Some important concepts with respect to concurrent game structures are specified as follows: For two states q and q' , q' is called a *successor* of q if there is a joint action $\vec{m} \in D(q)$ such that $q' = \delta(q, \vec{m})$. A *computation* of S is an infinite sequence $\lambda = q_0, q_1, q_2, \dots$ of states such that for all positions $i \geq 0$, the state q_{i+1} is a successor of the state q_i . We refer to a computation starting from state q as a q -computation. For a computation λ and a position $i \geq 0$, we use $\lambda[i]$, $\lambda[0, i]$, and $\lambda[i, \infty]$ to denote, respectively, the i th state of λ , the finite prefix q_0, q_1, \dots, q_i of λ , and the infinite suffix q_i, q_{i+1}, \dots of λ . A *strategy* for agent $a \in \Sigma$ is a function f_a that maps every nonempty finite state sequence $\lambda \in Q^+$ to an action such that if the last state of λ is q , then $f_a(\lambda) \in \{1, \dots, d_a(q)\}$.

²For the joint actions of an arbitrary agent set $A \subseteq Ag$, we always consider them as action vectors arranged in order of increasing IDs of the corresponding agents in A , instead of action sets.

And, the *outcomes* of a set of strategies F_A , called an *A-strategy*, one for each agent in $A \subseteq Ag$, from a state $q \in Q$ is the set $out(q, F_A)$ of computations, such that a computation $\lambda = q_0, q_1, q_2, \dots$ is in $out(q, F_A)$ if $q_0 = q$ and there is a joint action $\langle j_1, \dots, j_k \rangle \in D(q_i)$ such that (1) $j_a = f_a(\lambda[0, i])$ for all agents $a \in A$, and (2) $\delta(q_i, j_1, \dots, j_k) = q_{i+1}$.

The language of ATL \mathcal{L} is generated by the following grammar:

$$\varphi ::= p | \neg\varphi | \varphi_1 \vee \varphi_2 | \langle\langle A \rangle\rangle \circ \varphi | \langle\langle A \rangle\rangle \square \varphi | \langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2,$$

where $p \in \Pi$ is a proposition, and $A \subseteq Ag$ is a set of agents³.

As an abbreviation, we write $\langle\langle A \rangle\rangle \diamond \varphi$ for $\langle\langle A \rangle\rangle \top \mathcal{U} \varphi$.

We write $S, q \models \varphi$ to indicate that the formula φ holds at state q of a CGS S . When S is clear from the context, we write $q \models \varphi$. The relation \models is defined, for all states q of S , inductively as follows:

- For all $p \in \Pi$ we have $q \models p$ iff $p \in \pi(q)$.
- $q \models \neg\varphi$ iff $q \not\models \varphi$.
- $q \models \varphi_1 \vee \varphi_2$ iff $q \models \varphi_1$ or $q \models \varphi_2$.
- $q \models \langle\langle A \rangle\rangle \circ \varphi$ iff there exists a A -strategy, F_A , such that for all computations $\lambda \in out(q, F_A)$ we have $\lambda[1] \models \varphi$.
- $q \models \langle\langle A \rangle\rangle \square \varphi$ iff there exists a A -strategy, F_A , such that for all computations $\lambda \in out(q, F_A)$ and all positions $i \geq 0$, we have $\lambda[i] \models \varphi$.
- $q \models \langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there exists a A -strategy, F_A , such that for all computations $\lambda \in out(q, F_A)$ there exists a position $i \geq 0$ such that $\lambda[i] \models \varphi_2$ and for all positions $0 \leq j < i$ we have $\lambda[j] \models \varphi_1$.

2.2 Normative Systems

Given a concurrent game structure $S = \langle k, Q, \Pi, \pi, d, \delta \rangle$, a normative system (NS) is a function η such that

$$\eta(a, q) \subseteq \{1, \dots, d_a(q)\}$$

for all agents $a \in Ag$ and states $q \in Q$.

Intuitively, $\eta(a, q)$ is the set of “forbidden” (or “illegal”) actions for agent a in state q . The structure obtained from a CGS S by *implementing* an NS η , denoted as $S \uparrow \eta$, is the structure obtained from S by deleting all the forbidden actions. Note that, NS is defined as a *proper subset* of all the available actions to guarantee every agent will has at least one available actions in every state after having implemented an NS. Apparently, $S \uparrow \eta$ is still a concurrent game structure.

An existential and a universal sublanguage of ATL, denoted \mathcal{L}^e and \mathcal{L}^u , respectively, were defined in [14] by the following grammars ϵ and v respectively:

$$\epsilon ::= p | \epsilon \wedge \epsilon | \epsilon \vee \epsilon | \langle\langle Ag \rangle\rangle \circ \epsilon | \langle\langle Ag \rangle\rangle \square \epsilon | \langle\langle Ag \rangle\rangle \epsilon \mathcal{U} \epsilon$$

$$v ::= p | v \wedge v | v \vee v | \langle\langle \rangle\rangle \circ v | \langle\langle \rangle\rangle \square v | \langle\langle \rangle\rangle v \mathcal{U} v$$

where $p \in \Pi$.

Suppose we have a CGS S , an NS η , a state q in S , and formulas $\epsilon \in \mathcal{L}^e$, $v \in \mathcal{L}^u$. Then,

³We always assume that we are studying a fixed set Ag of agents and a fixed set Π of propositions. So, the language of ATL is a fixed set of formulas, and when we refer to “concurrent game structure” we actually mean a concurrent game structure with $|Ag|$ and Π as its components.

1. $S \dagger \eta, q \models \epsilon \Rightarrow S, q \models \epsilon$;
2. $S, q \models v \Rightarrow S \dagger \eta, q \models v$.

The first result tells us that the satisfaction of a \mathcal{L}^e formula cannot be *established* by implementing a NS⁴. The second result, in contrast, tells us that the satisfaction of a \mathcal{L}^u formula cannot be *avoided* by implementing a NS.

3. COALITIONAL NORMATIVE SYSTEMS

3.1 The Formal Framework

A coalitional normative system (CNS) for a concurrent game structure $S = \langle k, Q, \Pi, \pi, d, \delta \rangle$ is a tuple $\Gamma = \langle C, \vartheta \rangle$ with the following components:

- A coalition $C \subseteq \{1, \dots, k\}$.
- For each state $q \in Q$, a set $\vartheta(q) \subset D_C(q)$ of C -actions the agents in coalition C cannot collaboratively choose. The function ϑ is called coordination function.

Sometimes we call a CNS $\Gamma = \langle C, \vartheta \rangle$ as a C -norm. When CNSs are taken into consideration, certain joint action choices and computations will be ruled out. As in [17], we adopt the prefix “ Γ -conformant” to mean “permitted by Γ ”:

- A joint action $\bar{m} \in D(q)$ is called a Γ -conformant joint action iff $\nexists \bar{m}_C \in \vartheta(q)$ such that $\bar{m}|_C = \bar{m}_C$.
- A state q' is called a Γ -conformant successor of state q if there is a Γ -conformant joint action $\bar{m} \in D(q)$ such that $q' = \delta(q, \bar{m})$.
- A Γ -conformant computation of S is an infinite sequence $\lambda = q_0, q_1, q_2, \dots$ of states such that for all positions $i \geq 0$, the state q_{i+1} is a Γ -conformant successor of the state q_i .
- In each state $q \in Q$, an A -action \bar{m}_A for the agent set $A \subseteq Ag$, is called a Γ -conformant A -action in q iff $\exists \bar{m}_C \notin \vartheta(q)$ such that $\bar{m}_A|_A \cap C = \bar{m}_C|_A \cap C$.
- A set $F_A = \{f_a | a \in A\}$ of strategies, one for each agent in A , is called a Γ -conformant A -strategy iff for all nonempty finite state sequences $\lambda \in Q^+$, the A -action \bar{m}_A , given by F_A , is a Γ -conformant A -action.
- Finally, the Γ -conformant outcomes of a Γ -conformant A -strategy from a state $q \in Q$ is the set $out_\Gamma(q, F_A)$, such that, a Γ -conformant computation $\lambda = q_0, q_1, q_2, \dots$ is in $out_\Gamma(q, F_A)$ if $q_0 = q$ and there is a Γ -conformant joint action $\bar{m} \in D(q_i)$ such that (1) $j_a = f_a(\lambda[0, i])$ for all players $a \in A$, and (2) $\delta(q_i, \bar{m}) = q_{i+1}$.

Similarly, we can define the structure obtained from S by implementing a CNS $\Gamma = \langle C, \vartheta \rangle$, denoted as $S \dagger \Gamma$, as the structure obtained from S by deleting all the joint actions forbidden by Γ . Notice that, in most cases $S \dagger \Gamma$ is not an ordinary concurrent game structure any longer – agents in C will “discuss” in advance on which C -action should be selected, so a kind of “coalitional coordination” is explicitly represented in the structure.

⁴As result 1 is equivalent to $S, q \models \neg \epsilon \Rightarrow S \dagger \eta, q \models \neg \epsilon$.

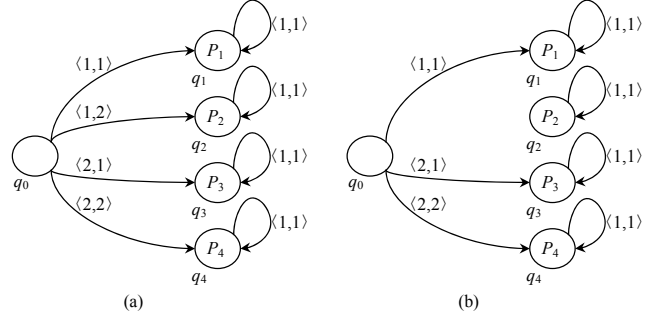


Figure 1: Implementing a CNS

CNS vs. NS: Coalitional normative system extends the concept of normative system by enabling selective restricting a coalition’s joint behavior. It is not difficult to discover that every NS is intrinsically a special CNS. Suppose that an action i of agent a is ruled out by an NS η , it actually means all the joint actions of the grand coalition that adopting i as a member are ruled out. So, for an arbitrary NS η we can always find an *equivalent* CNS $\Gamma_\eta = \langle Ag, \vartheta_\eta \rangle$ for the grand coalition, just let $\forall q \in Q : \vartheta_\eta(q) = \eta(1, q) \times \dots \times \eta(k, q)$. That is, the following result hold.

PROPOSITION 1. *Given a CGS S . For every NS η , there exists a CNS Γ such that $S \dagger \eta = S \dagger \Gamma$.*

But apparently there are some CNSs without equivalent NS. This means NS can be seen as a proper subset of CNS. Notice that although we can modify the original CGS in more ways by using CNSs, the resulting structure remains a CGS if and only if the CNS has an equivalent NS.

EXAMPLE 1. *Consider a CGS S depicted as Figure 1(a), that is, $S = \langle k, Q, \Pi, \pi, d, \delta \rangle$ where $k = 2$; $Q = \{q_0, \dots, q_4\}$; $\Pi = \{P_1, \dots, P_4\}$; $\pi(q_i) = P_i$ for all $1 \leq i \leq 4$; $d_1(q_0) = d_2(q_0) = 2$; $d_1(q_i) = d_2(q_i) = 1$ for all $1 \leq i \leq 4$; and $\delta(q_0, 1, 1) = q_1$; $\delta(q_0, 1, 2) = q_2$; $\delta(q_0, 2, 1) = q_3$; $\delta(q_0, 2, 2) = q_4$; $\delta(q_i, 1, 1) = q_i$ for all $1 \leq i \leq 4$. The following statements hold:*

1. An NS η such that $\eta(1, q_0) = \{1\}$; $\eta(2, q_0) = \{2\}$; $\eta(i, q_j) = \emptyset$ for all $i \in \{1, 2\}$ and $j \in \{1, \dots, 4\}$ has the same effect with the CNS $\Gamma = \langle C, \vartheta \rangle$ where $C = \{1, 2\}$; and $\vartheta(q_0) = \{\langle 1, 1 \rangle, \langle 1, 2 \rangle, \langle 2, 2 \rangle\}$; $\vartheta(q_i) = \emptyset$ for all $i \in \{1, \dots, 4\}$.
2. A CNS $\Gamma' = \langle C', \vartheta' \rangle$, where $C' = \{1, 2\}$; and $\vartheta'(q_0) = \{\langle 1, 2 \rangle\}$, $\vartheta'(q_i) = \emptyset$ for all $i \in \{1, \dots, 4\}$, can transform S to the structure depicted as Figure 1(b), but there doesn’t exist any NS which can achieve this transformation.
3. The structure depicted as Figure 1(b) cannot be modeled by any CGS.

3.2 Coordinated ATL

Consequently, in the presence of a CNS, we need to refine the interpretation for the ATL formulas. We call this new logic *Coordinated ATL* (CO-ATL), which directly inherits the ATL syntax but assumes slightly different semantics.

CO-ATL Semantics: We write $S, \Gamma, q \models \varphi$ to indicate that the formula φ holds at state q of a concurrent game structure S under the CNS Γ . When S and Γ is clear from the context, we write $q \models \varphi$. The relation \models is defined, for all states q of S , inductively as follows:

- For all $p \in \Pi$, we have $q \models p$ iff $p \in \pi(q)$.
- $q \models \neg\varphi$ iff $q \not\models \varphi$.
- $q \models \varphi_1 \vee \varphi_2$ iff $q \models \varphi_1$ or $q \models \varphi_2$.
- $q \models \langle\langle A \rangle\rangle \bigcirc \varphi$ iff there exists a Γ -conformant A -strategy, F_A , such that for all Γ -conformant computations $\lambda \in \text{out}_\Gamma(q, F_A)$ we have $\lambda[1] \models \varphi$.
- $q \models \langle\langle A \rangle\rangle \square \varphi$ iff there exists a Γ -conformant A -strategy, F_A , such that for all Γ -conformant computations $\lambda \in \text{out}_\Gamma(q, F_A)$ and all positions $i \geq 0$, we have $\lambda[i] \models \varphi$.
- $q \models \langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there exists a Γ -conformant A -strategy, F_A , such that for all Γ -conformant computations $\lambda \in \text{out}_\Gamma(q, F_A)$ there exists a position $i \geq 0$ such that $\lambda[i] \models \varphi_2$ and for all positions $0 \leq j < i$ we have $\lambda[j] \models \varphi_1$.

Model Checking CO-ATL: Because of its significance both in theory and practice, model checking is an important computational problem for any modal or temporal logic. The model checking problem for CO-ATL is defined as follows.

CO-ATL MODEL CHECKING:

Given: CGS $S = \langle k, Q, \Pi, \pi, d, \delta \rangle$, CNS $\Gamma = \langle C, \vartheta \rangle$, state $q \in Q$, and CO-ATL formula φ .

Question: Is $S, \Gamma, q \models \varphi$?

The lower bound of CO-ATL MODEL CHECKING is trivial, for we can reduce the ATL model checking problem, which is PTIME-complete [5], to it in polynomial time.

To show the upper bound, we can find a polynomial time algorithm for this problem. The algorithm is obtained from the ATL model checking algorithm proposed in [5] by rewriting the *Pre* function which given a set $A \subseteq \Sigma$ of agents and a set $\rho \subseteq Q$ of states, returns the set of states q such that from q the agents in A can cooperate and enforce the next state to lie in ρ . Formally, in the algorithm for CO-ATL, $\text{Pre}(A, \rho)$ contains state $q \in Q$ if there exists a Γ -conformant A -action \bar{m}_A in q such that for all Γ -conformant joint actions \bar{m} in q satisfying $\bar{m}|_A = \bar{m}_A$, we have $\delta(q, \bar{m}) \in \rho$. Finally, we can prove that this algorithm for CO-ATL is still a polynomial time algorithm.

THEOREM 2. CO-ATL MODEL CHECKING is PTIME-complete, and can be solved in time $\mathcal{O}(m \cdot l)$ for a CGS with m transitions and a CO-ATL formula φ of length l . The problem is PTIME-hard even for a fixed formula.

PROOF. We follow the steps of the proof for the ATL model checking complexity given by Alur *et al.* [5]. We reduce games played on CGSS with the constraint of a CNS to games played on turn-based synchronous game structures. The only difference is in building the corresponding 2-player turn-based synchronous game structure $S_A = \langle 2, Q_A, \Pi_A, \pi_A, \sigma_A, R_A \rangle$ with respect to a CGS S , an NS Γ , and a set of agents (players) $A \in \Sigma$. The components in S_A are defined as usual, but we have to redefine some related basic concepts based on the semantics of CO-ATL: For a state $q \in Q$, an A -move c at q is a Γ -conformant A -action defined

in this paper; An state $q' \in Q$ is a c -successor of q if there is a Γ -conformant joint action \bar{m} such that (1) $\bar{m}|_A = c$, and (2) $q' = \delta(q, \bar{m})$. It is easy to see that the aforementioned changes add no additional complexity to the structure of S_A , that is, if the original game structure S has m transitions, the turn-based synchronous structure S_C has $\mathcal{O}(m)$ states and transitions. This means we can find an algorithm for CO-ATL MODEL CHECKING which requires time $\mathcal{O}(m \cdot l)$. \square

So, with respect to the computational complexity of CO-ATL model checking we get the same result with that of ATL model checking.

CO-ATL vs. ATL: By adopting the empty CNS $\Gamma_\emptyset^C = \langle C, \vartheta_\emptyset \rangle$ where $\vartheta_\emptyset(q) = \emptyset$ for all $q \in Q$, we can identify the following relation between CO-ATL and ATL.

PROPOSITION 3. Given a CGS $S = \langle k, Q, \Pi, \pi, d, \delta \rangle$, then for all $q \in Q$, $C \subseteq \text{Ag}$ and $\varphi \in \mathcal{L}$, we have

$$S, \Gamma_\emptyset^C, q \models \varphi \Leftrightarrow S, q \models_{\text{ATL}} \varphi.$$

Remark that, compared with ATL, CO-ATL can represent and reason about α -abilities in a wider range of structures, that is, the class of structures obtained from concurrent game structures by implementing CNSs⁵. To differentiate between the two semantics, we use “ \models_{ATL} ” to denote the satisfaction relation in ATL.

3.3 Objectives and Effectiveness

We define the concept of objective for expressing the aim of the designer in CNS synthesis. Formally, an *atomic objective* is a state-formula pair, e.g. $\langle q, \varphi \rangle$, indicating that in state q the CO-ATL formula φ should be satisfied; Then *objectives* are generated by the following grammar o :

$$o ::= \langle q, \varphi \rangle \mid \neg o \mid o_1 \wedge o_2 \mid o_1 \vee o_2$$

where q is a state and φ is a CO-ATL formula. That is, an objective is a Boolean combination of atomic objectives which can express complex requirements about the system.

We adopt the expression “ $S \dagger \Gamma \rightsquigarrow o$ ” to mean the CNS Γ is *effective* for the objective o (in the CGS S), where the relation “ \rightsquigarrow ” is inductively defined as follows:

- $S \dagger \Gamma \rightsquigarrow \langle q, \varphi \rangle$ iff $S, \Gamma, q \models \varphi$;
- $S \dagger \Gamma \rightsquigarrow \neg o$ iff *not* $S \dagger \Gamma \rightsquigarrow o$;
- $S \dagger \Gamma \rightsquigarrow o_1 \wedge o_2$ iff $S \dagger \Gamma \rightsquigarrow o_1$ and $S \dagger \Gamma \rightsquigarrow o_2$;
- $S \dagger \Gamma \rightsquigarrow o_1 \vee o_2$ iff $S \dagger \Gamma \rightsquigarrow o_1$ or $S \dagger \Gamma \rightsquigarrow o_2$.

Intuitively, an effective CNS is a CNS that can fulfill the aim of the CNS designer, which is expressed as an objective.

3.4 Concepts of Optimality

Usually, there might be more than one effective CNSs for an objective. So it would be helpful if we can define some concepts of optimality for selecting among effective CNSs.

Minimality: The idea of minimality was firstly proposed in [6, 7], attempting to minimize the amount of constraints

⁵All concurrent game structures are in this class, because an arbitrary concurrent game structure S is also the structure obtained from S by implementing an empty CNS.

set on the agents and as such, capture the notion of maximal individual flexibility. We are going to transplant this idea to coalitional normative systems.

Given two CNSs $\Gamma_1 = \langle C, \vartheta_1 \rangle$ and $\Gamma_2 = \langle C, \vartheta_2 \rangle$. Γ_1 is said to be *less restrictive* than Γ_2 , denoted as $\Gamma_1 \leq \Gamma_2$, if and only if $\forall q \in Q \vartheta_1(q) \subseteq \vartheta_2(q)$. Γ_1 is said to be *strictly less restrictive* than Γ_2 , denoted as $\Gamma_1 < \Gamma_2$, if and only if $\Gamma_1 \leq \Gamma_2$ and $\exists q \in Q \vartheta_1(q) \subset \vartheta_2(q)$. And Γ is a *minimal* CNS if and only if $\nexists \Gamma'$ such that Γ' is effective and $\Gamma' < \Gamma$.

So a minimal CNS is one of the effective CNSs that put the least amount of constraints on the coalition.

Compactness: Notice that, in our framework the agents in the coalition have to negotiate with each other before selecting any joint actions. Basically, this process requires agents in the coalition sending messages to each other until an agreement on action selection has been reached. Obviously, more agents in the coalition means more complex the process is and more prone to cause error.

In this sense, with respect to an objective o if both $\Gamma = \langle C, \vartheta \rangle$ and $\Gamma' = \langle C', \vartheta' \rangle$ are effective coalitional normative systems, Γ is better than Γ' if C is a proper subset of C' (that is $C \subset C'$) – we say Γ is more *compact* than Γ' . An CNS $\Gamma = \langle C, \vartheta \rangle$ is a *compact* CNS if and only if it is an effective CNS and there doesn't exist any effective CNS Γ' which is more compact than Γ , and in this case, we say C is a *minimal controllable coalition*.

In other words, the key idea of compactness is minimizing the amount of agents in the coalition in order to minimize the communication cost in the system.

4. COALITIONAL NORMATIVE POWER AND ITS LIMITATION

Intuitively, the *normative power* of a coalition C is manifested by its ability of changing properties in CGSS by implementing C -norms. But very naturally the class of C -norms is not omnipotent, for we can show that some properties are inevitably beyond the reach of all the C -norms. In other words, the normative power of a coalition has its *limitations*. It is interesting to show what exactly the limitation is.

Power Limitation Characterization: For all formulas $\varphi \in \mathcal{L}$, we say φ 's *satisfaction cannot be established by (implementing) a C -norm* if and only if for all CNSs S , there doesn't exist any C -norm Γ , such that there is a $q \in Q$ satisfying $S, \Gamma_{\emptyset}^C, q \models \varphi$ and $S, \Gamma, q \not\models \varphi$; we say φ 's *satisfaction cannot be avoided by (implementing) a C -norm* if and only if for all CNSs S , there doesn't exist any C -norm Γ , such that there is a $q \in Q$ satisfying $S, \Gamma_{\emptyset}^C, q \models \varphi$ and $S, \Gamma, q \not\models \varphi$. Then the power limitation of the class of C -norm can be characterized by answering the following two questions:

1. *which fragment of \mathcal{L} is the set of formulas whose satisfaction cannot be established by a C -norm?*
2. *which fragment of \mathcal{L} is the set of formulas whose satisfaction cannot be avoided by a C -norm?*

We then define two fragments of the CO-ATL language, \mathcal{L}_C^+ and \mathcal{L}_C^- , which are generated by the grammars φ and ψ below respectively.

$$\varphi ::= p|\varphi_1 \wedge \varphi_2|\varphi_1 \vee \varphi_2|\langle\langle C^+ \rangle\rangle\langle\langle C^+ \rangle\rangle\langle\langle C^+ \rangle\rangle\varphi_1\mathcal{U}\varphi_2|\neg\psi$$

$$\psi ::= p|\psi_1 \wedge \psi_2|\psi_1 \vee \psi_2|\langle\langle C^- \rangle\rangle\langle\langle C^- \rangle\rangle\langle\langle C^- \rangle\rangle\psi|\langle\langle C^- \rangle\rangle\psi_1\mathcal{U}\psi_2|\neg\varphi$$

where $p \in \Pi$, $C \subseteq C^+ \subseteq Ag$, $\emptyset \subseteq C^- \subseteq Ag \setminus C$.

In the following, we will show that \mathcal{L}_C^+ and \mathcal{L}_C^- are exactly the answers to the above two questions respectively. That is, by \mathcal{L}_C^+ and \mathcal{L}_C^- we can *soundly* and *completely* characterize the the limitation of the normative power of coalition C .

Soundness and Completeness: First of all, we prove the following two lemmas, which implies that a C -norm cannot add any thing new to the strategic ability of a coalition that consists of a superset of C , and cannot avoid any strategic ability of a coalition that consists of a subset of $Ag \setminus C$.

LEMMA 4. *Given an arbitrary CGS S , a state q_0 in S and an arbitrary C -norm Γ . If C^+ is a set of agents satisfying $C \subseteq C^+ \subseteq Ag$ and F_{C^+} is a Γ -conformant C^+ -strategy, then there is a C^+ -strategy F'_{C^+} such that*

$$out(q_0, F'_{C^+}) = out_{\Gamma}(q_0, F_{C^+}).$$

PROOF. Always, we can define $F'_{C^+} = F_{C^+}$. And in all states q , after the agents in C^+ selected a C^+ -action, the available joint actions and available Γ -conformant joint actions for the agents in $Ag \setminus C^+$ are the same, that is, all the joint actions in $D_{Ag \setminus C^+}(q)$, as Γ put no constraint on the agents in $Ag \setminus C^+$. \square

LEMMA 5. *Given a CGS S , a state q_0 in S and an arbitrary C -norm Γ . If C^- is a set of agents satisfying $\emptyset \subseteq C^- \subseteq Ag \setminus C$ and F_{C^-} is a C^- -strategy, then there is an Γ -conformant C^- -strategy F'_{C^-} such that*

$$out_{\Gamma}(q_0, F'_{C^-}) \subseteq out(q_0, F_{C^-}).$$

PROOF. As Γ actually cannot put any constraints on the behavior of the agents in C^- , all the C^- -strategies are Γ -conformant joint strategies for C^- . So, we can define $F'_{C^-} = F_{C^-}$. And in all states q , for the agents in $Ag \setminus C^-$, the set of Γ -conformant joint actions is a subset of the joint actions, because of the effect of Γ . \square

Then, soundness of our characterization can be established by the following theorem.

THEOREM 6. *Given an arbitrary CGS S , an arbitrary C -norm Γ and an arbitrary state q in S . Then*

1. $\forall \varphi \in \mathcal{L}_C^+$, we have $S, \Gamma, q \models \varphi \Rightarrow S, \Gamma_{\emptyset}^C, q \models \varphi$.
2. $\forall \psi \in \mathcal{L}_C^-$, we have $S, \Gamma_{\emptyset}^C, q \models \psi \Rightarrow S, \Gamma, q \models \psi$.

PROOF. By induction on the structure of φ and ψ . For the case of propositions the conclusion trivially hold. For the other cases, suppose for all $\varphi \in \mathcal{L}_C^+$ and $\psi \in \mathcal{L}_C^-$ the conclusion holds. Then the satisfaction for the cases of $\varphi_1 \wedge \varphi_2$, $\psi_1 \wedge \psi_2$, $\varphi_1 \vee \varphi_2$, $\psi_1 \vee \psi_2$, $\neg\varphi$ and $\neg\psi$ are immediate.

Moreover, for all $C \subseteq C^+ \subseteq Ag$ and $\emptyset \subseteq C^- \subseteq Ag \setminus C$:

$S, \Gamma, q \models \langle\langle C^+ \rangle\rangle\langle\langle C^+ \rangle\rangle\langle\langle C^+ \rangle\rangle\varphi \Rightarrow$ (by the CO-ATL semantics) there is a Γ -conformant C^+ -strategy F_{C^+} such that for all Γ -conformant q -computations $\lambda \in out_{\Gamma}(q, F_{C^+})$, $S, \Gamma, \lambda[1] \models \varphi \Rightarrow$ (by the induction hypothesis) for all $\lambda \in out_{\Gamma}(q, F_{C^+})$, $S, \Gamma_{\emptyset}^C, \lambda[1] \models \varphi \Rightarrow$ (by lemma 4 and proposition 3) there is a F'_{C^+} such that for all $\lambda' \in out(q, F'_{C^+})$ we have $S, \lambda'[1] \models_{ATL} \varphi \Rightarrow$ (by the ATL semantics and proposition 3) $S, \Gamma_{\emptyset}^C, q \models \langle\langle C^+ \rangle\rangle\langle\langle C^+ \rangle\rangle\langle\langle C^+ \rangle\rangle\varphi$.

$S, \Gamma_{\emptyset}^C, q \models \langle\langle C^- \rangle\rangle\langle\langle C^- \rangle\rangle\langle\langle C^- \rangle\rangle\psi \Rightarrow$ (by proposition 3 and the ATL semantics) there is a C^- -strategy F_{C^-} such that for all $\lambda \in$

$out(q, F_{C^-}), S, \lambda[1] \models_{ATL} \psi \Rightarrow$ (by lemma 5) there is a Γ -conformant C^- -strategy F'_{C^-} for all $\lambda' \in out_{\Gamma}(q, F'_{C^-})$ we have $S, \lambda'[1] \models_{ATL} \psi \Rightarrow$ (by proposition 3 and the induction hypothesis) for all $\lambda' \in out_{\Gamma}(q, F'_{C^-})$ we have $S, \Gamma, \lambda'[1] \models \psi \Rightarrow$ (by the CO-ATL semantics) $S, \Gamma, q \models \langle\langle C^- \rangle\rangle \bigcirc \psi$.

Analogously, we can prove that the conclusion hold for $\langle\langle C^+ \rangle\rangle \square \varphi$, $\langle\langle C^+ \rangle\rangle \varphi_1 \mathcal{U} \varphi_2$, $\langle\langle C^- \rangle\rangle \square \psi$, and $\langle\langle C^- \rangle\rangle \psi_1 \mathcal{U} \psi_2$. \square

Finally, we can justify the completeness of our characterization by the following theorem.

THEOREM 7. (1) If $\varphi \notin \mathcal{L}_C^+$, then the satisfaction of φ can be established by a C -norm. (2) If $\varphi \notin \mathcal{L}_C^-$, then the satisfaction of φ can be avoided by a C -norm.

PROOF. We are only going to prove the first part of this theorem, as the proof for the second part is similar.

To show the satisfaction of all the formulas in $\mathcal{L} \setminus \mathcal{L}_C^+$ can be established by implementing a C -norm. Our method is mainly based on constructing the required concurrent game structure for each such formula.

Let $A \subseteq Ag$ and $A \cap C \neq C$ (i.e., A is not a C^+). For all formulas of the form $\langle\langle A \rangle\rangle \gamma$ where γ is of the form $\bigcirc \varphi$, $\square \varphi$, or $\varphi_1 \mathcal{U} \varphi_2$, and $\varphi, \varphi_1, \varphi_2$ are arbitrary CO-ATL formulas, we can construct a concurrent game structure S with a state q in its state space, satisfying $S, q \not\models_{ATL} \langle\langle A \rangle\rangle \gamma$, but $S, q \models_{ATL} \langle\langle A \cup C \rangle\rangle \gamma$. So, in such S , agents in the set $A \cup C$ have a joint strategy $F_{A \cup C}$ such that all computations in $out(q, F_{A \cup C})$ satisfy ψ . According to [5, 11], $F_{A \cup C}$ can be a set of “memory-free” strategies that map states to $A \cup C$ -actions. We construct the C -norm Γ to restrict the joint actions of the agents in set $A \cap C$ to only those are consistent with $F_{A \cup C}$. Then, by Γ , we have $S, \Gamma, q \models \langle\langle A \rangle\rangle \gamma$. That is, the satisfaction of $\langle\langle A \rangle\rangle \gamma$ can be established by implementing a C -norm. Proving the result “if $\varphi \notin \mathcal{L}_C^-$ then the satisfaction of $\neg \varphi$ can be established by implementing a C -norm” can be transformed to proving “if $\varphi \notin \mathcal{L}_C^-$ then the satisfaction of φ can be avoided by a implementing a C -norm”.

Moreover, it is straightforward to prove that if the satisfaction of φ can be established by a implementing C -norm, then the satisfaction of $\varphi \wedge \varphi'$, $\varphi \vee \varphi'$, $\langle\langle A \rangle\rangle \bigcirc \varphi$, $\langle\langle A \rangle\rangle \square \varphi$, $\langle\langle A \rangle\rangle \varphi' \mathcal{U} \varphi$, $\langle\langle A \rangle\rangle \varphi \mathcal{U} \varphi'$, $\neg \langle\langle A \rangle\rangle \bigcirc \varphi$, $\neg \langle\langle A \rangle\rangle \square \varphi$, $\neg \langle\langle A \rangle\rangle \varphi' \mathcal{U} \varphi$, and $\neg \langle\langle A \rangle\rangle \varphi \mathcal{U} \varphi'$, where $A \subseteq Ag$ and φ' is an arbitrary CO-ATL formula, can also be established by implementing a C -norm. \square

Interesting Corollaries: It is easy to show that all the CO-ATL formulas are beyond the normative power of an empty coalition, and the power limitation characterization for NS given by [14] is not complete.

COROLLARY 8. (1) The limitation of the normative power of coalition \emptyset can be characterized by $\mathcal{L}_{\emptyset}^+$ and $\mathcal{L}_{\emptyset}^-$, where both $\mathcal{L}_{\emptyset}^+$ and $\mathcal{L}_{\emptyset}^-$ are the class of CO-ATL formulas generated by the following grammar φ (i.e., all the CO-ATL formulas):

$$\varphi ::= p | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \langle\langle A \rangle\rangle \bigcirc \varphi | \langle\langle A \rangle\rangle \square \varphi | \langle\langle A \rangle\rangle \varphi_1 \mathcal{U} \varphi_2 | \neg \varphi$$

where $p \in \Pi$, and $\emptyset \subseteq A \subseteq Ag$.

(2) The limitation of the normative power of coalition Ag can be characterized by \mathcal{L}_{Ag}^+ and \mathcal{L}_{Ag}^- , which are the classes of CO-ATL formulas generated by the following grammars φ and ψ respectively:

$$\varphi ::= p | \varphi_1 \wedge \varphi_2 | \varphi_1 \vee \varphi_2 | \langle\langle Ag \rangle\rangle \bigcirc \varphi | \langle\langle Ag \rangle\rangle \square \varphi | \langle\langle Ag \rangle\rangle \varphi_1 \mathcal{U} \varphi_2 | \neg \psi$$

$$\psi ::= p | \psi_1 \wedge \psi_2 | \psi_1 \vee \psi_2 | \langle\langle \rangle\rangle \bigcirc \psi | \langle\langle \rangle\rangle \square \psi | \langle\langle \rangle\rangle \psi_1 \mathcal{U} \psi_2 | \neg \varphi$$

where $p \in \Pi$.

As the set of NSS is a strict subset of the class of Ag -norms. We can conclude that \mathcal{L}_{Ag}^+ and \mathcal{L}_{Ag}^- soundly characterize the limitation of the power of NSS. Although the completeness result doesn't hold for \mathcal{L}_{Ag}^+ and \mathcal{L}_{Ag}^- with respect to NSS, we are sure that \mathcal{L}_{Ag}^+ and \mathcal{L}_{Ag}^- is a more comprehensive characterization for the power limitation of NSS compared to \mathcal{L}^e and \mathcal{L}^u given by [14], because $\mathcal{L}^e \subset \mathcal{L}_{Ag}^+$ and $\mathcal{L}^u \subset \mathcal{L}_{Ag}^-$.

Moreover, we can compare the normative power of different coalitions. We say coalition C_1 is more powerful than coalition C_2 if and only if $\mathcal{L}_{C_1}^+ \subseteq \mathcal{L}_{C_2}^+$ and $\mathcal{L}_{C_1}^- \subseteq \mathcal{L}_{C_2}^-$. Then immediately we can show that for two arbitrary coalitions C_1 and C_2 if $C_1 \subseteq C_2$ then C_2 is more powerful than C_1 . Hence, with respect to normative power, \emptyset is the weakest coalition and Ag is the strongest coalition.

5. COMPLEXITY

5.1 Basic Computational Problems

The basic computational problems related to CNSs may include *checking whether a CNS is effective*, *checking whether there is an effective CNS*, and *finding an effective CNS*. We formalize them respectively as follows:

CNS EFFECTIVENESS:

Given: CGS S , CNS Γ and objective o .

Question: Is Γ effective for o ?

CNS FEASIBILITY:

Given: CGS S , coalition C and objective o .

Question: Is there a C -norm which is effective for o ?

CNS SYNTHESIS:

Given: CGS S , coalition C and objective o .

Output: A C -norm Γ that is effective for o .

Note that, similar problems have been proposed for NS in [14]. It has been established that the effectiveness problem for NS is in PTIME and the feasibility problem for NS is NP-complete. For CNS, we have the following results.

THEOREM 9. CNS EFFECTIVENESS is PTIME-complete, and can be solved in time $\mathcal{O}(m \cdot n \cdot l)$ for a CGS with m transitions, and an objective of length n , where the max length of the CO-ATL formulas in the objective is bounded by l .

PROOF. To see whether an objective is effective we can firstly determine the effectiveness of all the atomic objectives which requires time $\mathcal{O}(m \cdot l \cdot n)$, then the remain work equals verifying an assignment for a Boolean formula, which requires time $\mathcal{O}(n)$. So the overall time complexity is $\mathcal{O}(m \cdot l \cdot n)$. Thus this problem is in PTIME. With respect to the lower bound, PTIME-hardness is trivial, for verifying $S, \Gamma, q \models \varphi$ can be directly reduced to verifying $S \upharpoonright \Gamma \rightsquigarrow \langle q, \varphi \rangle$. \square

While CNS EFFECTIVENESS is tractable, CNS FEASIBILITY is possibly intractable according to the following theorem. Note that, our result is based on the ATL assumption that the agent number, i.e., k , is a constant. But the state number $|Q|$, and the max available action number of every agent in every state, i.e., d , are considered as variables.

THEOREM 10. CNS FEASIBILITY is NP-complete, even for concurrent game structures with only one agent.

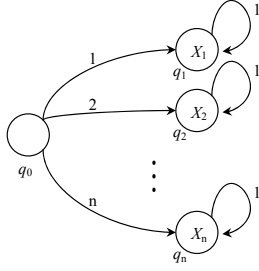


Figure 2: The reduction from SAT to CNS FEASIBILITY

PROOF. Membership in NP can be seen by the following nondeterministic algorithm:

- (1) guess a C -norm Γ ;
- (2) verify that Γ is effective for o .

Since step (1) can be done in non-deterministic polynomial time $\mathcal{O}(d^k \cdot |Q|)$, and step (2) requires only polynomial time.

To see NP-hardness we reduce SAT to it. Given a SAT instance $\phi(x_1, \dots, x_n)$. We create a CGS S depicted as Figure 2. That is, $S_\phi = \langle k, Q, \Pi, \pi, d, \delta \rangle$ where $k = 1$; $Q = \{q_0, \dots, q_{n+1}\}$; $\Pi = \{X_1, \dots, X_n\}$; $\pi(q_i) = \{X_i\}$ for all $i \in \{1, \dots, n\}$; $d_1(q_0) = \{1, \dots, n\}$, $d_1(q_i) = \{1\}$ for all $i \in \{1, \dots, n\}$; and $\delta(q_0, i) = q_i$ for all $i \in \{1, \dots, n\}$, $\delta(q_i, 1) = q_i$ for all $i \in \{1, \dots, n\}$. Let ϕ^* be the result of systematically substituting for every Boolean variable x_i in ϕ the CO-ATL expression $\langle\langle 1 \rangle\rangle \bigcirc X_i$. Then it is easy to see that ϕ is satisfiable if and only if there is a $\{1\}$ -norm which is effective for the objective $\langle q_0, \phi^* \rangle$. \square

The synthesis problem for CNS is a function problem that requires an answer more elaborate than “yes” or “no”.

THEOREM 11. CNS SYNTHESIS is FNP-complete, even for concurrent game structures with only one agent.

PROOF. Membership in FNP: Let L be the language for the CNS FEASIBILITY problem. By Theorem 10 we know that for all string x , to decide whether $x \in L$ is NP-complete. And we can define a relation R_L such that $R_L(x, y)$ if and only if $x \in L$ and y is an output of CNS SYNTHESIS given the instance x . It is easy to check that R_L is polynomial-time decidable and polynomially balanced.

To see FNP-hardness we reduce FSAT to it. The reduction is similar to that of Theorem 10. Given a boolean formula $\phi(x_1, \dots, x_n)$ we can create the CGS S_ϕ and the CO-ATL formula ϕ^* . Then we can see that x_1, \dots, x_n satisfy ϕ if and only if the CNS $\Gamma = \langle \{1\}, \vartheta \rangle$, where $\vartheta(q_0) = \{i \mid 0 \leq i \leq n \text{ and } x_i = 0\}$, $\vartheta(q_i) = \emptyset$ for all $i \in \{1, \dots, n\}$, is effective for the objective $\langle q_0, \phi^* \rangle$. \square

It is easy to see that the synthesis problem for NS is also FNP-complete. So now we can conclude that the effectiveness, feasibility and synthesis problems of CNS are no more complex than the corresponding problems of NS.

5.2 Complexity of Minimality Checking

The problem of Checking whether a CNS is a minimal CNS is a basic problem related to the concept of minimality.

MINIMAL CNS CHECKING:

Given: CGS S , CNS Γ and objective o .

Question: Is Γ a minimal CNS for o ?

THEOREM 12. MINIMAL CNS CHECKING is co-NP-complete.

PROOF. We can show that the complement problem to MINIMAL CNS CHECKING is NP-complete. That is, given a CGS S , a CNS Γ , and an objective o , determining whether there is a CNS such that $\Gamma' < \Gamma$ and Γ' is effective for o . Note that, an arbitrary CNS FEASIBILITY instance is an instance of this problem that taking Γ to be all the transitions in the CGS. So this problem subsumes CNS FEASIBILITY and thus is NP-hard. And the membership in NP is trivial. \square

5.3 Complexity of Compactness Checking

With respect to compactness, there are two basic decision problems, that is, *deciding whether a CNS is a compact CNS*, and *deciding whether a coalition is a minimal controllable coalition*. We define them formally as follows:

COMPACT CNS CHECKING:

Given: CGS S , CNS Γ and objective o .

Question: Is Γ a compact CNS for o ?

MINIMAL CONTROLLABLE COALITION CHECKING (MCC):

Given: CGS S , coalition C and objective o .

Question: Is C a minimal controllable coalition for o ?

THEOREM 13. COMPACT CNS CHECKING is co-NP-complete.

PROOF. The problem complement to COMPACT CNS CHECKING is as follows: given a CGS S , a CNS $\Gamma = \langle C, \vartheta \rangle$ and an objective o , is it true that Γ is not effective for o or there is a C' such that $C' \subset C$ and there is an effective C' -norm for o ? We can show this problem is NP-complete: NP-hardness is immediately, for it subsumes CNS FEASIBILITY; and since the amount of agents is a constant, the amount of subsets of C is bounded by a constant. So we can guess a CNS for every subset of C respectively in nondeterministic polynomial-time, and then verify that every CNS is an effective CNS in polynomial-time. This establishes the NP upper bound. \square

The problem of deciding whether a coalition is a minimal controllable coalition seems a harder problem, we can show that it is a problem that complete for the class DP⁶.

THEOREM 14. MINIMAL CONTROLLABLE COALITION CHECKING is DP-complete.

PROOF. Membership in DP can be seen from the following algorithm using an oracle for CNS FEASIBILITY:

- (1) query the oracle to see whether C is effective for o ;
- (2) query the oracle to see whether there is an agent $a \in C$ such that $C \setminus \{a\}$ is effective for o ;
- (3) if step (1) returns “yes” and step (2) returns “no” then return “yes”, otherwise return “no”.

To prove DP-hardness we reduce SAT-UNSAT [9] to it. Given a SAT-UNSAT instance $(\phi(x_1, \dots, x_m), \phi'(y_1, \dots, y_n))$, we can create a CGS S depicted as Figure 3.

Let Ψ be the formula $\langle\langle 1, 2 \rangle\rangle \bigcirc (P_X \wedge \phi^*) \vee (P_Y \wedge \phi'^*)$, where ϕ^* is the result of systematically substituting for every Boolean variable x_i in ϕ the CO-ATL expression $\langle\langle 1, 2 \rangle\rangle \bigcirc X_i$, and ϕ'^* is the result of systematically substituting for every Boolean variable y_i in ϕ' the CO-ATL expression $\langle\langle 1, 2 \rangle\rangle \bigcirc Y_i$. Then we can prove that (ϕ, ϕ') is a “yes” instance of SAT-UNSAT if

⁶DP is a complexity class “between” NP and PSPACE, and consists of all languages that are intersections of a language in NP and a language in co-NP (see [9] for the details of DP).

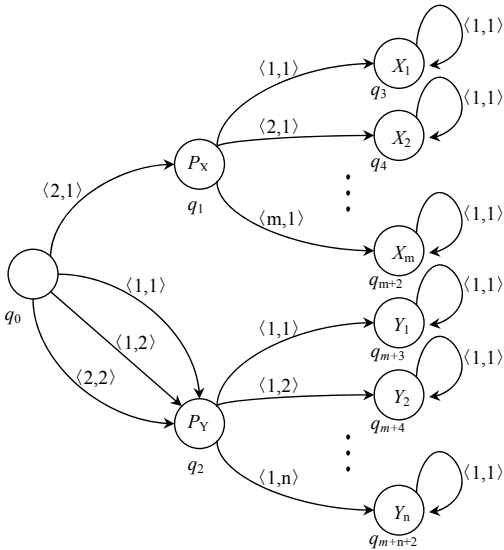


Figure 3: The reduction from SAT-UNSAT to MCC

and only if $\{1, 2\}$ is a minimal controllable coalition for the objective $\langle q_0, \Psi \rangle$.

For the \Rightarrow direction, we can define a $\{1, 2\}$ -norm that delete all the transitions from q_0 to q_2 , and delete some transitions started from q_1 to make Ψ satisfied in state q_0 . But for any $\{1\}$ -norm or $\{2\}$ -norm the transitions from q_0 to q_2 cannot be completely deleted at the same time, and since ϕ' is unsatisfiable, Ψ cannot be true in state q_0 .

For the \Leftarrow direction, the existence of effective $\{1, 2\}$ -norms requires ϕ or ϕ' is satisfiable. And the fact of Ψ cannot be satisfied in state q_0 by implementing any $\{1\}$ -norm or $\{2\}$ -norm means ϕ' is unsatisfiable, otherwise we can delete the transition from state q_0 to state q_1 and delete some transitions started from state q_2 by a $\{2\}$ -norm to make Ψ satisfied in state q_0 . So, ϕ is satisfiable and ϕ' is unsatisfiable. \square

6. CONCLUSIONS AND FUTURE WORK

We have proposed the framework for coalitional normative systems in this paper. Three aspects of theoretical work have been done: firstly, we have extended the semantics of ATL and proposed *Coordinated* ATL (CO-ATL) to support the formalizing of CNSS; secondly, we have proved that the limitation of the normative power of an arbitrary coalition C can be soundly and completely characterized by the CO-ATL fragments \mathcal{L}_C^+ and \mathcal{L}_C^- ; and thirdly, we have established the computational complexity of some key problems related to coalitional normative systems.

One opportunity for further research is to more systematically investigate the related computational complexity. As our current results are built on the conventional assumption that the amount of agents k is a constant. So it may be interesting to study the complexity when k is considered as a variable (as in [8]). Another possible further research is modeling the problem of finding an optimal CNS as an optimization problem (as the work in [2]).

7. ACKNOWLEDGEMENTS

This work was supported by the National Natural Science Foundation of China under Grant No.60503021,60721002 and 60875038; the Science and Technology Support Foundation of Jiangsu Province under Grant No.BE2009142 and BE2010180; and the Scientific Research Foundation of Graduate School of Nanjing University under Grant No.2011CL07. We are grateful to the reviewers for their helpful comments.

8. REFERENCES

- [1] T. Ågotnes, W. van der Hoek, M. Tennenholtz, and M. Wooldridge. Power in normative systems. In *Proceedings of AAMAS-09*, pages 145–152, 2009.
- [2] T. Ågotnes and M. Wooldridge. Optimal social laws. In *Proceedings of AAMAS-10*, pages 667–674, 2010.
- [3] T. Ågotnes, M. Wooldridge, and W. van der Hoek. Normative system games. In *Proceedings of AAMAS-07*, pages 876–883, 2007.
- [4] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. In *Proceedings of FOCS-97*, pages 100–109, 1997.
- [5] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.
- [6] D. Fitoussi and M. Tennenholtz. Minimal social laws. In *Proceedings of AAAI-98*, pages 26–31, 1998.
- [7] D. Fitoussi and M. Tennenholtz. Choosing social laws for multi-agent systems: Minimality and simplicity. *Artificial Intelligence*, 119:61–101, 2000.
- [8] W. Jamroga and J. Dix. Do agents make model checking explode (computationally)? In *Multi-Agent Systems and Applications IV (LNAI 3690)*, 2005.
- [9] C. H. Papadimitriou. The complexity of facets (and some facets of complexity). In *Proceedings of STOC-82*, pages 255–260, 1982.
- [10] M. Pauly and M. Wooldridge. Logic for mechanism design—a manifesto. In *Proceedings of the 2003 Workshop on Game Theory and Decision Theory in Agent Systems (GTDT-2003)*, 2003.
- [11] P. Schobbens. Alternating-time logic with imperfect recall. *Electronic Notes in Computer Science*, 85(2):82–93, 2004.
- [12] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *Proceedings of AAAI-92*, pages 276–281, 1992.
- [13] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: Off-line design. In *P.E. Agre and S.J. Rosenschein (Eds.), Computational Theories of Interaction and Agency*, pages 597–618, MIT Press, Cambridge, MA, 1996.
- [14] W. van der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: Effectiveness feasibility, and synthesis. *Synthese*, 156:1–19, 2007.
- [15] W. van der Hoek and M. Wooldridge. On the logic of cooperation and propositional control. *Artificial Intelligence*, 164:81–119, 2005.
- [16] M. Wooldridge, T. Agotnes, P. E. Dunne, and W. van der Hoek. Logic for automated mechanism design - a progress report. In *Proceedings of AAAI-07*, pages 9–16, 2007.
- [17] M. Wooldridge and W. van der Hoek. On obligations and normative ability: towards a logical analysis of the social contract. *J. of Applied Logic*, 3:396–420, 2005.

Practical Argumentation Semantics for Socially Efficient Defeasible Consequence

Hiroyuki Kido and Katsumi Nitta
Interdisciplinary Graduate School of Science and Engineering
Tokyo Institute of Technology, Japan
{kido, nitta}@ntt.dis.titech.ac.jp

ABSTRACT

An abstract argumentation framework and the semantics, often called Dungean semantics, give a general framework for nonmonotonic logics. In the last fifteen years, a great number of papers in computational argumentation adopt Dungean semantics as a fundamental principle for evaluating various kinds of defeasible consequences. Recently, many papers address problems not only with theoretical reasoning, i.e., reasoning about what to believe, but also practical reasoning, i.e., reasoning about what to do. This paper proposes a practical argumentation semantics specific to practical argumentation. This is motivated by our hypothesis that consequences of such argumentation should satisfy Pareto optimality because the consequences strongly depend on desires, aims, or values an individual agent or a group of agents has. We define a practical argumentation framework and two kinds of extensions, preferred and grounded extensions, with respect to each group of agents. We show that evaluating Pareto optimality can be translated to evaluating preferred extensions of a particular practical argumentation framework. Furthermore, we show that our semantics is a natural extension of Dungean semantics in terms of considering more than one defeat relation. We give a generality order of four practical argumentation frameworks specified by taking into account Dungean semantics and Pareto optimality. We show that a member of preferred extensions of the most specific one is not just Pareto optimal, but also it is theoretically justified.

Categories and Subject Descriptors

I.2.3 [Deduction and Theorem Proving]: Nonmonotonic reasoning and belief revision

General Terms

Theory

Keywords

Argumentation, Collective decision making, Reasoning, Logic-based approaches and methods

Cite as: Practical Argumentation Semantics for Socially Efficient Defeasible Consequence, Hiroyuki Kido and Katsumi Nitta, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 267-274.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

1. INTRODUCTION

An abstract argumentation framework and the semantics, often called Dungean semantics, give a general framework for nonmonotonic logics [6]. In the last fifteen years, a great number of papers in computational argumentation adopt Dungean semantics as a fundamental principle for evaluating states of arguments. Dungean semantics is defined on an abstract argumentation framework, denoted by AF , consisting of a set of arguments and a defeat relation on the set of arguments. Its main feature is that nonmonotonic reasoning can be realized without any internal structures of arguments such as languages or inferences. Recently, many papers address problems not only with theoretical reasoning, i.e., reasoning about what to believe, but also practical reasoning, i.e., reasoning about what to do, and apply Dungean semantics to these problems described as instances of AF or their expansions.

This paper shows that there exists a different kind of semantics specific to practical argumentation. Practical argumentation is known as the form of argumentation which aims at answering the question: ‘What is to be done [11]?’ The declaration is motivated by our hypothesis that decisions by practical argumentation must satisfy Pareto optimality. Consequences of practical argumentation are decisions of a course of action that an agent or a group of agents takes, and the decisions strongly depend on desires, aims, or values that it has. In such argumentation, agents are certain to avoid Pareto improvable decisions because if it is not Pareto optimal, there exists another decision that makes some agents better off and no one worse off. From this standpoint, there is no basis for believing that Dungean semantics gives an adequate principle for evaluating practical argumentation because it does not explain a relationship to social efficiency. The same holds true for the modification of Dungean semantics defined on a value-based argumentation framework [3]. Furthermore, many argument-based approaches for practical reasoning do not provide a sufficient explanation for applying Dungean semantics. In our view, Dungean semantics is specialized in evaluating acceptance of propositions as true, but it is insufficient for evaluating acceptance of actions as desirable.

In this paper, we propose practical argumentation semantics specific to practical argumentation. Practical argumentation semantics is defined on a practical argumentation framework consisting of a set of arguments without any internal structures, a set of agents, and a function from the set of agents to the power set of a binary relation on the set of arguments. The function outputs a defeat relation

that an inputted agent has. On the framework, we define two kinds of extensions, preferred and grounded extensions, with respect to each group of agents. In order to show the correctness of our theory, we show that evaluating Pareto optimality can be translated to evaluating preferred extensions of a particular practical argumentation framework. Furthermore, we show that evaluating defeasible consequences with Dungean semantics can also be translated to evaluating extensions of a particular practical argumentation framework. We give a generality order of four practical argumentation frameworks specified by taking into account Dungean semantics and Pareto optimality. We show that a member of preferred extensions of the most specific one is not just Pareto optimal, but also it is theoretically justified.

This paper is organized as follows. Section 2 shows a motivational example for addressing practical argumentation semantics. Section 3 gives preliminaries. In Section 4, we propose practical argumentation semantics, and in section 5, we show properties of the semantics. Section 6 gives an order relation of practical argumentation frameworks and Section 7 shows illustrative examples. Section 8 shows related works and Section 9 describes conclusions and future works.

2. MOTIVATIONAL EXAMPLE

Let us consider simple deliberative argumentation by which agents i and j try to decide what to do about buying an apartment. Agent i has concerns about safeness and quietness, and she prefers getting a safe neighborhood, avoiding an unsafe neighborhood, getting a quiet place, and avoiding a noisy place, in this order. In contrast, agent j has concerns about access to transportation, sunlight and safeness, and he prefers getting good access to transportation, avoiding bad access to transportation, getting a place with sufficient sunlight, and getting a safe neighborhood, in this order. Consider the following arguments put forward by agents i and j at some point in argumentation.

- A_i : We ought to buy apartment ‘ a ’ because it is located in a safe area.
- B_j : We ought to buy apartment ‘ b ’ because it is quiet and it has sufficient sunlight.
- C_j : We ought not to buy ‘ a ’ because it has bad access to transportation.
- D_i : We ought not to buy ‘ b ’ because the public security is poor and the access to transportation is bad.

What is the consequence of the argumentation? In other words, what actions would be taken by rational agents. We think that rational agents are certain to decide to take socially efficient actions. Pareto optimality is a formal criterion for evaluating efficiency, and a solution is Pareto optimal if no agents can be made better off without making someone else worse off. Our idea here is that we evaluate efficiency of practical argumentation semantics, proposed in this paper, by checking whether the consequences defined by the semantics are Pareto optimal or not. However, it is difficult to evaluate the above argumentation in terms of Pareto optimality because it differs completely from the problem setting that Pareto optimality assumes. It assumes that each agent has his/her individual preferences on outcomes

and implicitly assumes that any two distinct outcomes are incompatible. Our detailed idea is that we reduce the original argumentation to restricted ones that can be handled in a problem of Pareto optimality, and conclude that our semantics is efficient based on the fact that the consequences of the restricted argumentation are identical to Pareto optimal solutions. For example, consider the situation that they evaluate the arguments based on his/her own preference on the arguments. If we consider the restricted argumentation consisting of arguments A and B , then both A and B are Pareto optimal because agent i prefers A to B and agent j prefers B to A . If we consider the restricted argumentation consisting of arguments B and D , then only D is Pareto optimal because both agents prefer D to B . Our practical argumentation semantics must define defeasible consequences that are consistent with the evaluation of Pareto optimality in each restricted argumentation.

We have to take into account arguments about not only what to do, but also, what to believe in practical argumentation. Consider the following arguments put forward by agent j at the end of argumentation.

- E_j : ‘ a ’ is not located in a safe area because a murder occurred and the murderer is still at large.
- F_j : It takes five minutes from ‘ b ’ to the closest station and the station has two train lines. Further, there is a police office near the station. Therefore, the public security and the access to transportation are not bad.

In this situation, what is the consequence of the argumentation, or what actions would be taken by rational agents? A and D fail to justify their own actions because they cannot defeat the defeating arguments E and F , respectively. Therefore, the effects of these arguments on the decision should be canceled. We benefit from Dungean semantics for evaluating this kind of arguments, and combine our practical argumentation semantics and Dungean semantics in order to handle not only practically efficient, but also theoretically justified arguments.

3. PRELIMINARIES

Let G be a set and R be a binary relation on G , i.e., $R \subseteq G \times G$. R is called reflexive if $(x, x) \in R$, for all $x \in G$, transitive if whenever $(x, y) \in R$ and $(y, z) \in R$ then $(x, z) \in R$, for all $x, y, z \in G$, and antisymmetric if whenever $(x, y) \in R$ and $(y, x) \in R$ then $x = y$, for all $x, y \in G$. R is called quasi-order if it is reflexive and transitive, and partial order if it is reflexive, transitive, and antisymmetric. The inverse relation of R , denoted by R^{-1} , and the complement relation of R , denoted by \bar{R} , are defined as $R^{-1} = \{(x, y) \mid (y, x) \in R\}$ and $\bar{R} = \{(x, y) \mid (x, y) \notin R\}$, respectively. The inverse complement relation of R is the complement relation of the inverse relation of R , i.e., $\overline{R^{-1}}$.

Welfare economics is a branch of economics that is concerned with the evaluation of alternative economic situations (states, configurations) from the point of view of the society’s well being [10]. One of the prominent measures for evaluating society’s well being is Pareto optimality defined as follows.

Definition 1. An outcome $o_1 \in \mathcal{O}$ is Pareto optimal (or Pareto efficient) if there is no other outcome $o_2 \neq o_1$ such that $\forall i \in I, o_2 \succeq_i o_1$ and $\exists j \in I, o_2 \succ_j o_1$.

In other words, a solution is Pareto optimal if no agents can be made better off without making someone else worse off.

The abstract argumentation framework [6] is one of the argument-based approaches for nonmonotonic reasoning. Its main feature is that nonmonotonicity arises from the interactions between conflicting arguments, not in the process of constructing arguments. The abstract argumentation framework is especially abstract because it takes no account of the internal structures of arguments and only takes account of the external structures between arguments, i.e., defeat relation. The framework allows us to define various semantical notions of argumentation extensions. These notions are intended to capture various types of nonmonotonic consequence. The basic formal notions, with some terminological changes, are as follows.

Definition 2. [6] The abstract argumentation framework is defined as a pair $AF = \langle AR, defeat \rangle$ where AR is a set of arguments, and defeat is a binary relation on AR , i.e. $defeat \subseteq AR \times AR$.

- A set S of arguments is said to be conflict-free if there are no arguments A, B in S such that A defeats B .
- An argument $A \in AR$ is acceptable with respect to a set S of arguments iff for each argument $B \in AR$: if B defeats A then B is defeated by an argument in S .
- A conflict-free set of arguments S is admissible iff each argument in S is acceptable with respect to S .
- A preferred extension of an argumentation framework AF is a maximal (with respect to set inclusion) admissible set of AF .

For argumentation framework AF , an argument is justified with respect to AF if it is in every preferred extension of AF , and is defensible with respect to AF if it is in some but not all preferred extensions of AF [13].

4. PRACTICAL ARGUMENTATION SEMANTICS

Practical argumentation semantics is a general rule for defining notions of defeasible consequences of a practical argumentation. Practical argumentation is known as the form of argumentation which aims at answering the question: ‘What is to be done [11]?’ Practical argumentation as shown in Section 2 handles two different kinds of arguments. One is the argument concluding actions that a group of agents should do or should not do, and the other is the argument concluding truth of propositions. We call these two kinds of arguments practical and theoretical arguments, respectively. In this paper, we assume that a set $Args$ of arguments is divided into a set $Pargs$ of practical arguments and a set $Targs$ of theoretical arguments where $Args = Pargs \cup Targs$ and $Pargs \cap Targs = \emptyset$ hold. The assumption is based on the observation that these two kinds of arguments should be formally distinguished not at the level of abstract arguments without any internal structures of arguments, but at the level of internal structures of arguments such as logical languages or inferences. We define a practical argumentation framework as follows.

Definition 3. A practical argumentation framework, denoted by $PRAF$, is a pair $PRAF = \langle Args, Agents \rangle$,

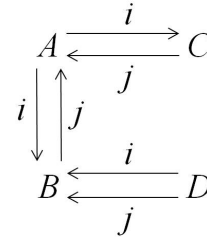


Figure 1: Arguments and subjective defeat relations

Defeat \succ , where $Args$ is a set of arguments, $Agents$ is a set of agents, and *Defeat* is a function that maps $Agents$ into $2^{Args \times Args}$.

PRAF characteristically has each agent i 's defeat relation defined by $Defeat(i)$. This reflects the fact that defeat relations between practical arguments are subjective because they strongly depend on preferences, desires, aims, values, morality, or ethics that an individual agent has. The individual agent's defeat relation might be substantiated by subjective preferences, values, and/or ethics, objective logical contradiction, or any combination thereof. *PRAF* abstracts any such internal information about arguments, and it consists of a minimal number of elements that practical argumentation semantics can be defined. In what follows, we say that x defeats y under i if there exist $i \in Agents$ and $(x, y) \in Defeat(i)$.

Example 1. The following is the practical argumentation framework consisting of some arguments and defeat relations shown in Section 2.

$$\begin{aligned} PRAF &= \langle \{A, B, C, D\}, \{i, j\}, Defeat \rangle \\ Defeat(i) &= \{(A, B), (A, C), (D, B)\} \\ Defeat(j) &= \{(B, A), (C, A), (D, B)\} \end{aligned}$$

The arguments and the defeat relations can be shown in Figure 1. There exists an arrow from x to y with label i if x defeats y under i .

In what follows, we assume an arbitrary but fixed practical argumentation framework. Consequences of practical argumentation are decisions of a course of action that an agent or a group of agents takes. Therefore, the consequences must be consistent. One of the properties that a set of arguments has is conflict-freeness.

Definition 4. A set $S \subseteq Args$ of arguments is conflict-free to a set $N \subseteq Agents$ of agents if for all arguments $A, B \in S$, A does not defeat B under any agent $i \in N$.

We define a notion of acceptability. The basic idea of acceptability is that a set N of rational agents would accept an argument A if each argument defeating A under some agent is defeated by some argument under an agent in N .

Definition 5. An argument $A \in Args$ is acceptable to a set $N \subseteq Agents$ of agents with respect to a set $S \subseteq Args$ of arguments if each argument defeating A under an agent $i \in Agents$ is defeated by an argument $B \in S$ under an agent $j \in N$.

In contrast to acceptable arguments defined in Dungean semantics, our acceptable arguments differ from one set of agents to another. Note that acceptability does not require that each argument defeating A is defeated by an argument $B \in S$ under all agents $j \in N$. The notion of admissibility is defined on the basis of conflict-freeness and acceptability.

Definition 6. A set $S \subseteq \text{Args}$ of arguments is admissible to a set $N \subseteq \text{Agents}$ of agents if S is conflict-free to N and each argument in S is acceptable to N with respect to S .

Self-admissibility is defined in this paper. Intuitively, every argument A in a self-admissible set can defeat every argument defeating A by A itself. In other words, A can defend itself without relying on any other arguments.

Definition 7. A set $S \subseteq \text{Args}$ of arguments is self-admissible to a set $N \subseteq \text{Agents}$ of agents if S is conflict-free to N and each argument $A \in S$ is acceptable to N with respect to $\{A\}$.

We call an element of a self-admissible set a self-admissible argument. Note that it is not always true that a self-admissible set has only one element. The credulous or preferred semantics of a practical argumentation framework is defined by the notion of preferred extension.

Definition 8. A set $S \subseteq \text{Args}$ of arguments is a preferred extension to a set $N \subseteq \text{Agents}$ of agents if S is a maximal admissible set to N .

The credulous semantics provides defeasible consequences of a practical argumentation framework. Another defeasible consequence of a practical argumentation framework is provided by a skeptical or grounded semantics. The semantics is defined by using the following operator.

Definition 9. Let $S \subseteq \text{Args}$ and $N \subseteq \text{Agents}$. Then the operator F^N for N is defined as follows.

- $F^N(S) = \{A \in \text{Args} \mid A \text{ is acceptable to } N \text{ with respect to } S\}$

Definition 10. A set of $S \subseteq \text{Args}$ of arguments is a grounded extension to a set $N \subseteq \text{Agents}$ of agents if S is the least fixed point of F^N .

Example 2. Both $\{A, D\}$ and $\{C, D\}$ are preferred extensions to $\{i, j\}$, and $\{D\}$ is a grounded extension of $\{i, j\}$ in Example 1.

5. PROPERTIES OF PRACTICAL ARGUMENTATION SEMANTICS

In this section, we aim to show the relationships between our practical argumentation semantics and both Pareto optimality and Dungean semantics. For Pareto optimality, we show that evaluating Pareto optimal solutions can be translated to evaluating preferred extensions of a particular practical argumentation framework. The following lemma shows the relationship between preferred extensions and self-admissible arguments.

Lemma 1. Let $\text{PRAF} = \langle \text{Args}, \text{Agents}, \text{Defeat} \rangle$ be a practical argumentation framework where the complement of $\text{Defeat}(i)$ is transitive, for all $i \in \text{Agents}$. An argument $A \in \text{Args}$ is a member of some preferred extension to Agents iff A is self-admissible to Agents .

PROOF. (\Leftarrow) From Definition 8, a preferred extension is a conflict-free admissible set. Thus, if $\{A\}$ is admissible set to Agents then there exists a preferred extension S to Agents such that $\{A\} \subseteq S$. (\Rightarrow) We show that the contradiction is derived under the assumptions that A is a member of some preferred extension S to Agents and A is not self-admissible to Agents . Under the assumptions, there exists an argument $B \in \text{Args}$ defeating A , under an agent $i \in \text{Agents}$, that is not defeated by A under any agent $j \in \text{Agents}$ and is defeated by a third argument $C \in S$ under an agent $k \in \text{Agents}$. Formally, the following formulas hold for S .

$$\begin{aligned} & \exists B \in \text{Args} (\exists i \in \text{Agents} ((B, A) \in \text{Defeat}(i)) \\ & \wedge \forall j \in \text{Agents} ((A, B) \notin \text{Defeat}(j)) \\ & \wedge \exists k \in \text{Agents} \exists C \in S ((C, B) \in \text{Defeat}(k))) \\ \Rightarrow & \exists B \in \text{Args} \exists i \in \text{Agents} ((B, A) \in \text{Defeat}(i)) \\ & \wedge \exists j \in \text{Agents} \exists C \in S ((A, B) \notin \text{Defeat}(j) \\ & \wedge (C, B) \in \text{Defeat}(j)) \quad (1) \\ \Rightarrow & \exists B \in \text{Args} \exists i \in \text{Agents} ((B, A) \in \text{Defeat}(i)) \\ & \wedge \exists j \in \text{Agents} \exists C \in S ((C, A) \in \text{Defeat}(j)) \quad (2) \end{aligned}$$

(2) can be derived from (1) under the following assumption that the complement of $\text{Defeat}(i)$ is transitive.

$$\begin{aligned} & \forall A, B, C \in \text{Args} \forall i \in \text{Agents} ((A, B) \notin \text{Defeat}(i) \\ & \wedge (C, A) \notin \text{Defeat}(i) \rightarrow (C, B) \notin \text{Defeat}(i)) \\ \Leftrightarrow & \forall A, B, C \in \text{Args} \forall i \in \text{Agents} ((A, B) \notin \text{Defeat}(i) \\ & \wedge (C, B) \in \text{Defeat}(i) \rightarrow (C, A) \in \text{Defeat}(i)) \end{aligned}$$

$A, C \in S$ and there exists $j \in \text{Agents}$ such that $(C, A) \in \text{Defeat}(j)$ in (2). This contradicts the assumption that S is conflict-free to Agents . \square

In Lemma 1, $\text{Defeat}(i)$ is assumed to be transitive. In Theorem 1, $\text{Defeat}(i)$ is substituted by the inverse complement of i 's preference expressed as quasi-order. The transitivity in Lemma 1 is a minimal assumption that makes Lemma 1 hold. The following lemma shows the relationship between self-admissible arguments and Pareto optimal solutions.

Lemma 2. Let \mathcal{O} be a set of outcomes, Agents be a set of agents, and \succsim_i ($i \in \text{Agents}$) be a quasi-order on \mathcal{O} . An outcome $o \in \mathcal{O}$ is Pareto optimal with respect to each agent i 's preference \succsim_i iff o is self-admissible of $\text{PRAF} = \langle \mathcal{O}, \text{Agents}, \text{Defeat} \rangle$ to Agents where $\text{Defeat}(i) = \mathcal{Z}_i$, for all $i \in \text{Agents}$.

PROOF. $o \in \mathcal{O}$ is self-admissible to Agents iff the following formula holds.

$$\begin{aligned} & \nexists i \in \text{Agents} (o \mathcal{Z}_i o) \wedge \forall o_1 \in \mathcal{O} (\exists i \in \text{Agents} \\ & (o_1 \mathcal{Z}_i o) \rightarrow \exists j \in \text{Agents} (o \mathcal{Z}_j o_1)) \quad (3) \end{aligned}$$

(3) can be transformed to the following formulas based on

the assumption that \succsim_i is a quasi-order.

$$\begin{aligned}
& \forall o_1 \in \mathcal{O} (\exists i \in \text{Agents}(o_1 \succsim_i o) \rightarrow \exists j \in \text{Agent} \\
& \quad (o \succsim_j o_1)) \\
\Leftrightarrow & \forall o_1 \in \mathcal{O} (\exists i \in \text{Agents}(o_1 \succ_i o \vee o \succsim_i o_1 \wedge o_1 \succsim_i o) \\
& \quad \rightarrow \exists j \in \text{Agents}(o \succsim_j o_1)) \\
\Leftrightarrow & \forall o_1 \in \mathcal{O} (\exists i \in \text{Agents}(o_1 \succ_i o) \vee \exists k \in \text{Agents} \\
& \quad (o \succsim_k o_1 \wedge o_1 \succsim_k o) \rightarrow \exists j \in \text{Agents}(o \succsim_j o_1)) \\
\Leftrightarrow & \forall o_1 \in \mathcal{O} ((\exists i \in \text{Agents}(o_1 \succ_i o) \rightarrow \exists j \in \text{Agents} \\
& \quad (o \succsim_j o_1)) \wedge (\exists k \in \text{Agents}(o \succsim_k o_1 \wedge o_1 \succsim_k o) \rightarrow \\
& \quad \exists l \in \text{Agents}(o \succsim_l o_1))) \\
\Leftrightarrow & \forall o_1 \in \mathcal{O} (\exists i \in \text{Agents}(o_1 \succ_i o) \rightarrow \exists j \in \text{Agents} \\
& \quad (o \succsim_j o_1)) \\
\Leftrightarrow & \forall o_1 \in \mathcal{O} (\nexists i \in \text{Agents}(o_1 \succ_i o) \vee \exists j \in \text{Agents} \\
& \quad (o \succsim_j o_1)) \\
\Leftrightarrow & \nexists o_1 \in \mathcal{O} (\exists i \in \text{Agents}(o_1 \succ_i o) \wedge \forall j \in \text{Agents} \\
& \quad (o \not\succsim_j o_1)) \tag{4}
\end{aligned}$$

(4) is equivalent to the definition of Pareto optimality, and therefore, o is Pareto optimal. \square

From Lemma 1 and Lemma 2, we can reach the following theorem.

Theorem 1. Let \mathcal{O} be a set of outcomes, Agents be a set of agents, and \succsim_i ($i \in \text{Agents}$) be a quasi-order on \mathcal{O} . An outcome $o \in \mathcal{O}$ is Pareto optimal with respect to each agent i 's preference \succsim_i iff o is a member of some preferred extension of $\text{PRAF} = \langle \mathcal{O}, \text{Agents}, \text{Defeat} \rangle$ to Agents where $\text{Defeat}(i) = \succsim_i$, for all $i \in \text{Agents}$.

Theorem 1 shows that evaluating Pareto optimal solutions can be translated to evaluating preferred extensions of a particular practical argumentation framework. This fact provides a theoretical basis for concluding that the practical argumentation semantics credulously justifies Pareto optimal solutions. Note that due to the particularity of the practical argumentation framework, it is generally the case that evaluating preferred extensions cannot be translated to evaluating Pareto optimal solutions.

For Dungean semantics, a link exists between our practical argumentation semantics and Dungean semantics.

Proposition 1. Let $AF = \langle \text{Args}, \text{defeat} \rangle$ be an abstract argumentation framework. The preferred extensions and the grounded extension of AF are equivalent to the preferred extensions and the grounded extension of $\text{PRAF} = \langle \text{Args}, \text{Agents}, \text{Defeat} \rangle$ to Agents where $\text{Agents} = \{i\}$ and $\text{Defeat}(i) = \text{defeat}$.

Proposition 1 shows that our practical argumentation semantics justifies defeasible consequences instead of Dungean semantics. Furthermore, it provides a theoretical basis for concluding that our practical argumentation semantics is a natural extension of Dungean semantics in terms of handling subjective defeat relations. Note that due to the particularity of the practical argumentation framework, it is generally the case that evaluating extensions of a practical argumentation framework cannot be translated to evaluating extensions of an abstract argumentation framework.

6. GENERALITY ORDER FOR PRACTICAL ARGUMENTATION FRAMEWORKS

This section gives a generality order of four practical argumentation frameworks specified by taking into account Dungean semantics and Pareto optimality. A practical argumentation framework and our practical argumentation semantics are insufficient to handle the practical argumentation shown in Section 2 because it takes no account of theoretical arguments that play a role of evaluating the truth of statements in practical arguments. Hence, we take into account theoretical arguments and the defeat relations that are unrelated to agents' subjective preferences, desires, values, morality, and ethics. A possible way to handle theoretical evaluation in practical argumentation is to unify our practical argumentation semantics and Dungean semantics into one semantics. However, it does not always work well. We sometimes take an attitude that reasoning about beliefs should be skeptical while reasoning about action should be credulous [12]. A unified semantics cannot evaluate these two types of reasoning in different ways, i.e., by preferred or grounded semantics. We take a different approach that stratifies a practical argumentation framework by taking into account an abstract argumentation framework evaluated by Dungean semantics. In addition, we further stratify the framework by considering Pareto optimality.

Definition 11. Let $AF = \langle \text{Args}, \text{defeat} \rangle$ be an abstract argumentation framework where $\text{Args} = \text{Targs} \cup \text{Pargs}$ and $\text{defeat} \subseteq \text{Targs} \times \text{Args}$, and $\text{PRAF} = \langle S, \text{Agents}, \text{Defeat} \rangle$ be a practical argumentation framework where $S \subseteq \text{Args}$.

1. PRAF is a justified practical argumentation framework with respect to AF , denoted by JPRAF , if all arguments in S are members of the grounded extension of AF .
2. PRAF is a practical argumentation framework for Pareto optimality, denoted by PRAF_{PO} , if the complement of $\text{Defeat}(i)$ is quasi-order, for all $i \in \text{Agents}$.
3. PRAF is a justified practical argumentation framework for Pareto optimality, denoted by JPRAF_{PO} , if PRAF is a justified practical argumentation framework with respect to AF and PRAF is a practical argumentation framework for Pareto optimality.

AF does not allow practical arguments to defeat any arguments while it allows theoretical arguments to defeat theoretical and practical arguments. Figure 2 shows a generality order of practical argumentation frameworks in Definition 11. Top of the order is a general argumentation framework and bottom of the order is the most specialized practical argumentation framework, i.e., JPRAF_{PO} . Note that it is generally the case that the intersection of the grounded extension of AF and the union of all preferred extensions of PRAF to a set of agents is not equal to the union of all preferred extensions of JPRAF to the set of agents. It means that we cannot obtain the same consequences with the preferred extensions of JPRAF to a set of agents by parallel evaluation of the grounded extension of AF and the preferred extensions of PRAF to the set of agents. From Theorem 1, an argument $A \in \text{Args}$

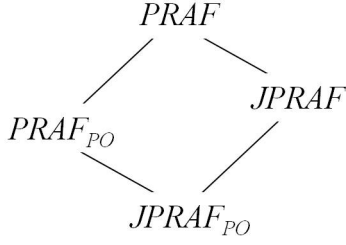


Figure 2: The generality order of practical argumentation frameworks

is a member of some preferred extension of $PRAF_{PO} = \langle Args, Agents, Defeat \rangle$ to $Agents$ iff A is Pareto optimal with respect to each agent i 's preference defined by the inverse complement of $Defeat(i)$. $JPRAF_{PO}$ is $PRAF_{PO}$. Therefore, it is noteworthy that a member of preferred extensions of $JPRAF_{PO}$ is not just Pareto optimal but also it is theoretically justified with respect to AF .

7. ILLUSTRATIVE EXAMPLES

This section shows illustrative examples of specialized practical argumentation frameworks and consequences of the frameworks. We make the specialized frameworks by restricting a general practical argumentation framework. Restriction is defined as follows.

Definition 12. Let $PRAF = \langle Args, Agents, Defeat \rangle$ be a practical argumentation framework. The restriction of $PRAF$ to $S \subseteq Args$ is the practical argumentation framework $PRAF \downarrow_S = \langle S, Agents, Defeat' \rangle$ where $Defeat'(i) = Defeat(i) \cap (S \times S)$ for all $i \in Agents$.

Consider the set $Pargs = \{A, B, C, D, E, F\}$ of practical arguments and the set $Targs = \{G, H\}$ of theoretical arguments. Each argument states that we ought to buy apartment 'a' because it is located in a safe area, denoted by an argument A , we ought to buy apartment 'b' because it is quiet and it has good access to transportation, by B , we ought to buy apartment 'c' because it has good access to transportation, by C , we ought not to buy 'a' because it is beyond the budget, by D , we ought not to buy 'b' because it is beyond the budget and located in a unsafe area, by E , we ought not to buy 'c' because it does not have sufficient sunlight, by F , 'b' is not located in a safe area because an airstrip is now under construction in that area, by G , and we can buy 'a' within the budget because the real estate gives us discount, by H . Furthermore, the objective defeat relation $defeat = \{(G, B), (H, D)\}$ and the following subjective defeat relations are given.

$$\begin{aligned} Defeat(i) &= \{(A, B), (A, C), (A, D), (B, C), (E, B)\} \\ Defeat(j) &= \{(B, A), (C, A), (C, F), (D, A), (E, B), \\ &\quad (F, C)\} \end{aligned}$$

Figure 3 shows these arguments and the objective and subjective defeat relations where the filled arrows depict the objective defeat relations. Consider following abstract argumentation framework AF and practical argumentation

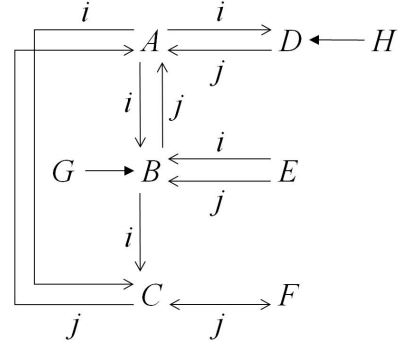


Figure 3: The whole defeat relations between arguments

framework $PRAF$.

$$\begin{aligned} AF &= \langle Targs \cup Pargs, defeat \rangle \\ PRAF &= \langle Pargs, Agents, Defeat \rangle \end{aligned}$$

The preferred extension, and the grounded extension as well, of AF is $\{A, C, E, F, G, H\}$. Moreover, the preferred extensions of $PRAF$ to $\{i, j\}$ are $\{A, E, F\}$ and $\{C, D, E\}$, and the grounded extension of $PRAF$ to $\{i, j\}$ is $\{E\}$. The following is a justified practical argumentation framework with respect to AF obtained by restricting $PRAF$ to $\{A, C, E, F\}$.

$$\begin{aligned} JPRAF &= \langle \{A, C, E, F\}, \{i, j\}, Defeat_{JPRAF} \rangle \\ Defeat_{JPRAF}(i) &= \{(A, C)\} \\ Defeat_{JPRAF}(j) &= \{(C, A), (C, F), (F, C)\} \end{aligned}$$

$\{C\}$ is the grounded extension of $JPRAF$ to $\{i, j\}$ and both $\{A, E, F\}$ and $\{C, E\}$ are the preferred extensions of $JPRAF$ to $\{i, j\}$. Following $PRAF_{PO}$ is a practical argumentation framework for Pareto optimality obtained by restricting $PRAF$ to $\{A, B, C\}$.

$$\begin{aligned} PRAF_{PO} &= \langle \{A, B, C\}, \{i, j\}, Defeat_{PRAF_{PO}} \rangle \\ Defeat_{PRAF_{PO}}(i) &= \{(A, B), (A, C), (B, C)\} \\ Defeat_{PRAF_{PO}}(j) &= \{(B, A), (C, A)\} \end{aligned}$$

The grounded extension of $PRAF_{PO}$ to $\{i, j\}$ is the empty set and the preferred extensions of $PRAF_{PO}$ to $\{i, j\}$ are $\{A\}$ and $\{B\}$. Therefore, both A and B are Pareto optimal arguments with respect to agents' preferences defined by the inverse complements of $Defeat_{PRAF_{PO}}(x)$, for $x = i, j$. Note that these inverse complements are quasi-order. Following $JPRAF_{PO}$ is a justified practical argumentation framework with respect to AF for Pareto optimality obtained by restricting $PRAF$ to $\{A, C\}$.

$$\begin{aligned} JPRAF_{PO} &= \langle \{A, C\}, \{i, j\}, Defeat_{JPRAF_{PO}} \rangle \\ Defeat_{JPRAF_{PO}}(i) &= \{(A, C)\} \\ Defeat_{JPRAF_{PO}}(j) &= \{(C, A)\} \end{aligned}$$

The grounded extension of $JPRAF_{PO}$ to $\{i, j\}$ is the empty set and the preferred extensions of $JPRAF_{PO}$ to $\{i, j\}$ are $\{A\}$ and $\{C\}$. Therefore, both A and C are not just Pareto optimal but also they are theoretically justified with respect to AF .

8. RELATED WORK

Deliberation is a type of dialogue in which a group of agents or a single agent tries, through looking at a set of alternatives, to make a decision about which course of action among the possible alternatives to take [18]. Our practical argumentation semantics can be applied to the evaluation of argument-based deliberation. Many argument-based approaches for deliberation or practical reasoning, however, apply Dungean semantics as a fundamental principle for evaluating arguments. For instance, a decision of a single agent's course of action, who has more than one desire, is formalized by instances of an abstract argumentation framework [18, 12]. In [18], the authors propose two kinds of practical reasoning, positive and negative practical syllogisms, denoted by *PPS* and *NPS*. They are incorporated into arguments for drawing desirable and undesirable actions, respectively. Dungean semantics is used for evaluating arguments, and consequently decides what the best action is. In [12], the author gives a combined formalization for skeptical epistemic reasoning interleaved with credulous practical reasoning. He distinguishes practical arguments from theoretical arguments by informally dividing logical formulas into epistemic and practical ones. Epistemic and practical arguments are evaluated by skeptical semantics and credulous semantics defined by Dungean semantics, respectively. On the other hand, these approaches do not discuss the relationship to efficiency. We think that a decision of a course of action and the notion of efficiency are inseparable even when single agent's argumentation.

In [16], the authors introduce seven dialectical inference rules on dialectical logic DL and weaker dialectical logic DM [15] in order to realize concession or compromise from inconsistent theory. They apply the inferences into argument-based negotiation for reaching agreement. Similarly, in [9], the authors propose compromise reasoning on an abstract lattice, and illustrate that compromise arguments incorporating the reasoning realize compromise-based justification. Furthermore, in [1], the authors propose an abstract framework for argument-based negotiation, and introduce the notion of concession as an essential element of negotiation. We think that concessions and compromises should be chosen from Pareto optimal solutions. However, none of them discuss the relationship between Pareto optimality with the notions of concession and compromise.

Recently, in [14], the authors analyze Dungean semantics by means of Pareto optimality. Pareto optimal solutions are defined based on each agent's preferences on extensions of an abstract argumentation framework. However, it does not provide new argumentation semantics that is consistent with Pareto optimality. In [8], the authors introduce Pareto optimality into argument-based negotiation. The notion, however, is used in a process of negotiation, and it is not evaluated by argumentation semantics.

From the point of view of argumentation semantics, some authors introduce nonclassical semantics such as stage semantics [17], semi-stable semantics [4], ideal semantics [7], CF2 semantics [2], and prudent semantics [5] on Dung's abstract argumentation framework. All of them intend to overcome or improve some limitations or drawbacks of Dungean semantics. On the other hand, our practical argumentation semantics is defined on the different framework, i.e., practical argumentation framework consisting of minimal number of elements that our semantics can be defined. Further-

more, it specializes in evaluating practical argumentation, and it does not address the improvement of Dungean semantics. In order to evaluate practical argumentation involving agents' values, the author proposes value-based argumentation frameworks, denoted by *VAF*, and modifies Dungean semantics [3]. The modified semantics corresponds to applying Dungean semantics to each abstract argumentation framework constructed from an individual agent's defeat relation. The paper, however, does not explain the relationship between the modified semantics with another theory. We think that it is essential for establishing the correctness of the modified semantics.

9. CONCLUSIONS AND FUTURE WORK

We proposed a practical argumentation semantics specific to practical argumentation. This attempt was motivated by our hypothesis that extensions of practical argumentation are certain to be efficient in terms of Pareto optimality. We showed that an outcome is Pareto optimal iff the outcome is a member of some preferred extension of a particular practical argumentation framework. This fact established that our practical argumentation semantics is efficient in terms of Pareto optimality. We showed that our practical argumentation semantics is a natural extension of Dungean semantics in terms of handling more than one defeat relation. We defined four ordered practical argumentation frameworks and gave illustrative examples of these frameworks by restricting the most general one. We need to formalize dialectical proof theory for our semantics, i.e., procedures determining whether an argument is a member of some extension or not. In particular, we are interested in formalizing proof theory of *JPRAF* that need to be calculated based on two semantics, Dungean semantics and our semantics.

10. REFERENCES

- [1] L. Amgoud, Y. Dimopoulos, and P. Moraitis. A general framework for argumentation-based negotiation. In *Proc. of The 4th International Workshop on Argumentation in Multi-Agent Systems (ArgMAS 2007)*, pages 1–17, 2008.
- [2] B. Baroni. Scc-recursiveness: ageneral schema for argumentation semantics. *Artificial Intelligence*, 168(1-2):162–210, 2005.
- [3] T. J. M. Bench-Capon. Value-based argumentation frameworks. In *Proc. of The 9th International Workshop on Non-Monotonic Reasoning (NMR 2002)*, pages 443–454, 2002.
- [4] M. Caminada. Semi-stable semantics. In *Proc. of The First International Conference on Computational Models of Argument (COMMA 2006)*, pages 121–130, 2006.
- [5] S. Coste-Marquis, C. Devred, and P. Marquis. Prudent semantics for argumentation frameworks. In *Proc. of The 17th International Conference on Tools with Artificial Intelligence (ICTAI 2005)*, pages 568–572, 2005.
- [6] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming, and n -person games. *Artificial Intelligence*, 77:321–357, 1995.
- [7] P. M. Dung, P. Mancarella, and F. Toni. A dialectic procedure for sceptical, assumption-based

- argumentation. In *Proc. of The First International Conference on Computational Models of Argument (COMMA 2006)*, pages 145–156, 2006.
- [8] P. M. Dung, P. M. Thang, and F. Toni. Towards argumentation-based contract negotiation. In *Proc. of The Second International Conference on Computational Models of Argument (COMMA 2008)*, pages 134–146, 2008.
- [9] H. Kido and M. Kurihara. Computational dialectics based on specialization and generalization: a new reasoning method for conflict resolution. In *Proc. Second International Workshop on Juris-informatics (JURISIN 2008)*, pages 228–241, 2009.
- [10] A. Koutsoyiannis. *Modern Microeconomics*. Palgrave Macmillan, 1979.
- [11] E. Lagerspetz. Ad hominem arguments in practical argumentation. *Argumentation*, 9(2):363–370, 1995.
- [12] H. Prakken. Combining sceptical epistemic reasoning with credulous practical reasoning. In *Proc. of The First International Conference on Computational Models of Argument (COMMA 2006)*, pages 311–322, 2006.
- [13] H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-classical Logics*, 7:25–75, 1997.
- [14] I. Rahwan and K. Larson. Pareto optimality in abstract argumentation. In *Proc. of The 23rd National Conference on Artificial Intelligence*, pages 150–155, 2008.
- [15] R. Routley and R. K. Meyer. Dialectical logic, classical logic, and the consistency of the world. *Studies in East European Thought*, 16(1-2):1–25, 1976.
- [16] H. Sawamura, M. Yamashita, and Y. Umeda. Applying dialectic agents to argumentation in e-commerce. *Electronic Commerce Research*, 3(3-4):297–313, 2003.
- [17] B. Verheij. Two approaches to dialectical argumentation: admissible sets and argumentation stages. In *Proc. 8th Dutch Conference on Artificial Intelligence (NAIC'96)*, pages 357–368, 1996.
- [18] D. Walton. *Argumentation methods for artificial intelligence in law*. Springer, 2005.

Taming the Complexity of Linear Time BDI Logics

Nils Bulling
Clausthal University of Technology
bulling@in.tu-clausthal.de

Koen V. Hindriks
Delft University of Technology
k.v.hindriks@tudelft.nl

ABSTRACT

Reasoning about the mental states of agents is important in various settings, and has been recognized as vital for teamwork. But the complexity of some of the more well-known agent logics that facilitate reasoning about mental states prohibits the use of these logics in practice. An alternative is to investigate fragments of these logics that have a lower complexity but are still expressive enough for reasoning about the mental states of (other) agents. We explore this alternative and take as our starting point the linear time variant of BDI logic ($\mathbf{BDI}_{\text{LTL}}$). We summarize some of the relevant known complexity results for e.g. \mathbf{LTL} , $\mathbf{KD45}_n$, and $\mathbf{BDI}_{\text{LTL}}$ itself. We present a tableau-based method for establishing complexity bounds, and provide a map of the complexity of (various fragments of) $\mathbf{BDI}_{\text{LTL}}$. Finally, we identify a few fragments that may be usefully applied for reasoning about mental states.

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*

General Terms

Theory, Verification

Keywords

reasoning about mental states, linear time BDI logic, satisfiability, complexity

1. INTRODUCTION

In a social context, and more specifically for teamwork, the ability of an agent to reason about other agents has been recognized as vital [7]. In particular, reasoning about ones own and the mental states of other agents is important to be successful in such contexts. Reasoning about the mental states of others is needed to establish joint commitments and joint intentions [17, 4], collaboration and cooperation [17], teamwork [4, 7, 20], and coordination more generally [8].

Cite as: Taming the Complexity of Linear Time BDI Logics, Nils Bulling and Koen V. Hindriks, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 275–282.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

One application area is socio-cognitive robotics and human-robot teamwork [15], where modeling of social behavior is needed and it has been recognized that “consideration of the knowledge, abilities, goals, and even feelings of others” is required.

In practice, however, little use is made of logical approaches such as [4, 18, 9] that support reasoning about mental states due to the inherent complexity of the logics. For example, the satisfiability problem for *agent dynamic logic*, a logic closely related to the KARO framework, is in $\mathbf{2EXPTIME}$ [19] and for the TEAMLOG framework including group attitudes as well as propositional dynamic logic it is $\mathbf{EXPTIME}$ -complete [9]. Practical approaches typically do not maintain many of the formal properties of mental attitudes, and, as noted in [11], “the need for high-level logic-based languages capturing the key components of the BDI model remains.” This problem has also motivated the work reported in [9]. More generally, the issue is related to “the gap between theory and practice” [21] since complexity bounds at least theoretically determine what can be done in principle to bridge this gap. In practice, agent platforms typically have been restricted to reason *with* the beliefs and goals of a single agent [2, 3] and do not allow reasoning *about* the beliefs and goals of other agents, e.g. having a belief about the goal of another agent. Identifying such extended fragments therefore is important as it may allow the use of logical fragments in practice for *reasoning about other agents*.

One approach to deal with this problem is to study and identify fragments of agent logics that have a lower complexity but still support reasoning about other agents’ mental states. There are general techniques for identifying such fragments and reducing the complexity. For example, it is known that restricting the number of propositional atoms used or the depth of modal nesting may reduce the complexity of a modal logic [12]. It is not cognitively plausible either that humans use unlimited depth of reasoning [7].

The problem we explore in this paper is the satisfiability problem of fragments of $\mathbf{BDI}_{\text{LTL}}$. We present a tableau-based proof method for a family of $\mathbf{BDI}_{\text{LTL}}$. The tableau-method can also be used to analyze the complexity of this family and we show $\mathbf{BDI}_{\text{LTL}}$ is in \mathbf{PSPACE} . We then explore which fragments have a significantly lower complexity than full linear time BDI logic. More specifically, the aim is to identify fragments for which the satisfiability problem is in \mathbf{NP} . Identifying such fragments is a first step towards establishing reasonable computational performance as typical problem instances may be easier to solve [13], and, as even satisfiability for propositional logic is an \mathbf{NP} -complete prob-

lem we cannot do better without restricting this underlying logic. Our main motivation for doing so is that agents need a tool for reasoning about other agents’ mental states in order to coordinate their actions and a logic-based approach seems most suitable. We also briefly informally consider the expressivity of these fragments.

The paper is organized as follows. Section 2 reviews relevant related work. Section 3 briefly introduces $\mathbf{BDI}_{\mathbf{LTL}}$ and discusses some fragments that are promising from a computational point of view. In Section 4 a tableau-based method for proving satisfiability is introduced. This method provides the basis for some of the complexity results for $\mathbf{BDI}_{\mathbf{LTL}}$. Section 5 then presents the main complexity results for $\mathbf{BDI}_{\mathbf{LTL}}$ and for fragments of the logic. Section 6 informally discusses whether some minimal requirements are met for these fragments to be useful for reasoning about other agents. Finally, Section 7 concludes the paper.

2. RELATED WORK AND RESULTS

Although significant work has been done in isolating fragments of various modal logics including \mathbf{LTL} [5], logics of knowledge and belief [13], and combinations thereof [6] for which the satisfiability problem is in \mathbf{NP} , as far as we know, no existing work has identified fragments with similar complexity that allow the combination of *informational* and *motivational* attitude operators with *time*. However, if we want our agents to reason about both the informational as well as the motivational states of other agents we need exactly this combination. We believe that incorporating time is essential to be able to differentiate various types of goals such as achievement and maintenance goals (cf. also Section 6).¹

Here we build on the work of [18] which introduces (a family of) linear time BDI logic(s) $\mathbf{BDI}_{\mathbf{LTL}}$. $\mathbf{BDI}_{\mathbf{LTL}}$ provides a logical framework that allows an agent to distinguish between different mental attitudes, i.e. beliefs versus desires/intentions, and between different types of desires/intentions by means of temporal operators. This sets our work apart from [9] where complexity issues of a multi-agent logic called TEAMLOG are investigated in a setting without time. Before we explore fragments of $\mathbf{BDI}_{\mathbf{LTL}}$ itself, we first review relevant complexity results for linear temporal logic and logics of mental attitudes available in the literature. This will be useful for identifying fragments of $\mathbf{BDI}_{\mathbf{LTL}}$ for which the satisfiability problem is in \mathbf{NP} . In the remainder we assume that Φ denotes a *set of propositions*.

Normal modal logics. A starting point for our search for a computational logic for reasoning about the mental states of (other) agents is provided by the extensive work on logics of knowledge and belief reported [13]. [13] presents results that show that the complexity of the satisfiability problem for single agent logics of knowledge $\mathbf{S5}$ and belief $\mathbf{KD45}$ are \mathbf{NP} -complete, for multi-agent logics of knowledge $\mathbf{S5}_n$ and belief $\mathbf{KD45}_n$ are \mathbf{PSPACE} -complete, and extensions with a common knowledge operator are $\mathbf{EXPTIME}$ -complete.

For conative logics that are used for modeling the motivational attitudes of agents typically the modal logic \mathbf{KD} is used [21]. The logic \mathbf{KD} is in between \mathbf{K} and $\mathbf{S4}$, i.e. $\mathbf{K} \subseteq \mathbf{KD} \subseteq \mathbf{S4} = \mathbf{KT4}$. According to Ladner’s Theorem [1] this means that the satisfiability problem for \mathbf{KD} is \mathbf{PSPACE} -complete.

¹We will sometimes also talk about *goals* if there is no need to differentiate between desires and intentions.

In [9] it is shown that combining the multi-agent logic of belief and the multi-agent conative logic does not increase the complexity of the satisfiability problem which remains \mathbf{PSPACE} -complete.

Restricted settings for the logics $\mathbf{K}, \mathbf{K45}, \mathbf{KD45}, \mathbf{S5}, \mathbf{S4}$, and their multi-agent versions are considered in [12] and [9]. The main results are that satisfiability checking can be done in *linear-time* for many standard multi-agent extensions of \mathbf{K} by *bounding the number of propositional atoms and the depth of modal operators*. Here, we will just present the result for $\mathbf{KD45}_n$ which are of interest for our purposes. It is shown that satisfiability checking can be done in *linear time* if the number of propositions and nestings is fixed. The problem remains in this class for the single-agent setting ($n = 1$) even if nestings are not bounded. The problem gets harder if there is no restriction on the number of propositions and only a bound on the number of nestings. In this case satisfiability checking is \mathbf{NP} -complete. [9] shows that by bounding the modal depth by a constant the satisfiability problem of combinations of multi-agent belief and multi-agent conative logic is also \mathbf{NP} -complete. Only bounding the number of propositional atoms does not lower complexity. Bounding *both* the number of propositional atoms *and* the depth of modal operators reduces complexity to linear time, however. This is true even when group attitudes such as common belief and collective intentions are added [9]. We use $depth^m(\varphi)$ to denote the *modal depth* of φ (i.e. the number of nested modal operators); e.g. $K_i p \wedge K_i K_j q$ as modal depth 2. (Cf. Section 3.2 for a formal definition.) In the following table we use $md \leq c$, $c \in \mathbb{N}_0$, to denote the restriction to formulae φ with $depth^m(\varphi) \leq c$. We use $|\Phi|$ to denote the number of propositions. The first cell of a row is understood as a constraint, e.g. $|\Phi| \leq c$, $md \leq c'$ characterises the case in which the number of propositions and the modal depth is bounded by some natural numbers c and c' , respectively. In the table below we summarize some complexity results of the satisfiability problem which are relevant for our study:

	$\mathbf{K45}, \mathbf{KD45}, \mathbf{S5}$	$\mathbf{K}, \mathbf{KD}_n, \mathbf{K45}_n, \mathbf{KD45}_n, \mathbf{S5}_n$
no constraints	\mathbf{NP} -compl.	\mathbf{PSPACE} -compl.
$ \Phi \leq c$	linear time	\mathbf{PSPACE} -compl.
$md \leq c$	\mathbf{NP} -compl.	\mathbf{NP} -compl.
$ \Phi \leq c, md \leq c'$	linear time	linear time

Temporal Logics. It is well-known that the complexity of the satisfiability problem for \mathbf{LTL} is \mathbf{PSPACE} -complete and for \mathbf{CTL} $\mathbf{EXPTIME}$ -complete. In [5] a large number of propositional fragments of \mathbf{LTL} is considered and complexity results for both model checking and satisfiability are established. Restrictions on the temporal operators that may be used, the number of propositional atoms, and the temporal depth are considered. Satisfiability for a limited number of fragments turns out to be in the class \mathbf{NP} . This is shown for the following fragments of \mathbf{LTL} : (1) The “future-only” fragment; (2) The “next-time” fragment; and (3) the fragment of \mathbf{LTL} that allows no nesting of temporal operators. Satisfiability for the fragment with a fixed number of propositional atoms *and* limited temporal depth (but no restrictions on the temporal operators) can be solved in deterministic logarithmic space \mathbf{L} . The table below summarises the relevant results. $\mathbf{LTL}(\mathcal{U})$ is used to denote the fragment with \mathcal{U} being the only temporal operator. Similarly to the modal depth, we define the *temporal depth*, $depth^t(\varphi)$, of φ . In this case the number of nested *temporal* operators is considered. $td \leq c$ is also defined and used analogously to md .

	LTL	LTL(\mathcal{U})
$ \Phi \leq c, td \leq c'$	L	L
$td=0$	NP-compl.	NP-compl.
$ \Phi = 1$	PSPACE-compl.	P
$td = 2$	PSPACE-compl.	PSPACE-compl.

Logics of Knowledge and Time. It is natural to combine linear time with combinations of other modal operators (for representing mental attitudes) but only work on combining belief/knowledge and time is known to us. In [14] results related to numerous logics of knowledge and time are considered. A result of this work is that agents that do not forget or do not learn greatly increasing the complexity of reasoning about knowledge and time. It is shown that combining the logic of knowledge and linear time results in a logic for which the satisfiability problem is **PSPACE**-complete and the satisfiability problem for the combination of knowledge and branching time is **EXPTIME**-complete.

3. LINEAR TIME BDI LOGIC

In the literature surveyed, the complexity of fragments that combine time, and multi-agent belief and motivational attitudes is not discussed. Of course, closely related work does exist, and some of the more important work has been discussed above. We take the work reported in [18] as our starting point, which introduces a linear time BDI logic and presents decision methods for satisfiability using a tableau-based approach. Below, we first define the language and its semantics and then continue to discuss potentially interesting fragments from a complexity point of view.

3.1 Language and Semantics

The logic **BDI_{LTL}** introduced here is based more or less on [18]. We use the same language but extend it with multiple modal operators to be able to represent the mental attitudes of multiple agents. The semantics has also been set up slightly differently, and we use runs and time points as the basis for accessibility relations instead of worlds that are related through states in [18]; our setup is similar to that in [10].

The language of **BDI_{LTL}** includes temporal operators, and multiple belief, desire, and intention operators, one for each agent out of a finite set of agents. The temporal operators are the usual linear time temporal operators for next time and the until operator.

DEFINITION 1 (THE LANGUAGE OF **BDI_{LTL}).** *Let Φ be a set of propositional atoms with typical element p and Agt be a finite set of agents with typical element i :*

$$\varphi \in \mathcal{L} ::= p \mid \neg\varphi \mid \varphi \wedge \psi \mid \varphi \mathcal{U} \psi \mid \bigcirc\varphi \mid B_i\varphi \mid D_i\varphi \mid I_i\varphi$$

Models for the language \mathcal{L} of **BDI_{LTL}** consist of runs and of indexed relations on these runs used to define the semantics of mental attitudes of agents. Runs are derived from a given set of states and a transition relation on those states and are used to interpret temporal operators.

A run r over a set of states Q is an infinite sequence from Q^ω . We use $r[i]$ to denote the i th state on run r , starting from $i = 0$. $r[i, \infty]$ is used to denote the sub-run of r that starts at i . That is, $r[i, \infty] = r[i]r[i+1]\dots$. Given a set of states Q and a transition relation \rightarrow , the set of all runs induced by (Q, \rightarrow) is denoted by $\mathcal{R}_{(Q, \rightarrow)}$. $\mathcal{R}_{(Q, \rightarrow)}$ thus consists of all runs r for which we have that: $\forall i \in \mathbb{N}_0 : r[i] \rightarrow r[i+1]$. Finally, the set of *time points* (r, m) given a set \mathcal{R} of runs is defined by: $\text{Points}_{\mathcal{R}} = \{(r, m) \mid r \in \mathcal{R}, m \in \mathbb{N}_0\}$.

DEFINITION 2 (MODELS FOR **BDI_{LTL}).** *A model for the language of **BDI_{LTL}** is a tuple: $\mathcal{M} = (Q, \rightarrow, \mathcal{R}, \{\mathcal{B}_i\}_{i \in \text{Agt}}, \{\mathcal{D}_i\}_{i \in \text{Agt}}, \{\mathcal{I}_i\}_{i \in \text{Agt}}, \pi)$ where*

- Q is a non-empty set of states,
- $\rightarrow \subseteq Q \times Q$ a serial (temporal) accessibility relation,
- $\mathcal{R} \subseteq \mathcal{R}_{(Q, \rightarrow)}$ is a non-empty set of runs,
- $\mathcal{B}_i \subseteq \text{Points}_{\mathcal{R}} \times \text{Points}_{\mathcal{R}}$ is a transitive, serial, and Euclidean belief accessibility relation,
- $\mathcal{D}_i \subseteq \text{Points}_{\mathcal{R}} \times \text{Points}_{\mathcal{R}}$ is a desire accessibility relation,
- $\mathcal{I}_i \subseteq \text{Points}_{\mathcal{R}} \times \text{Points}_{\mathcal{R}}$ is called an intention accessibility relation, and
- $\pi : Q \rightarrow \mathcal{P}(\Phi)$ a labelling or valuation function.

Models for **BDI_{LTL}** include the usual restrictions on the type of accessibility relations in the definition of a model. We are also interested in some additional restrictions that may be imposed on models. For example, \mathcal{B} may also be reflexive, to obtain the usual semantics for *knowledge*, and we consider the additional constraint where \mathcal{D} is serial to exclude inconsistent desires. It is well-known that these constraints correspond with particular axiom schema labeled **K**, **T**, **D**, **4**, **5**. This gives rise to a family of BDI logics and we introduce some notation to refer to different variants. We use **BDI_{LTL}** to refer to **BDI_{LTL}^{KD45,K,KD}**, i.e. the logic with **KD45** belief operators, **K** desire operators, and **KD** intention operators. Alternatively, by varying the restrictions on the accessibility relations, we obtain, for example, the logic **BDI_{LTL}^{S5,KD,KD}** which combines knowledge, consistent desires, and intentions. In the following we define $L := \{\mathbf{K}, \mathbf{KD}, \mathbf{KD45}, \mathbf{S4}, \mathbf{S5}\}$ and often write **BDI_{LTL}^{X,Y,Z}** where we implicitly assume that $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \in L$.

It is usual to consider a range of additional constraints on models, which give, for example, rise to interaction axioms [9, 18]. For reasons of space, we consider only two of the more interesting interaction axioms. That is, we consider the constraint $\mathcal{D}_i \subseteq \mathcal{B}_i$ giving rise to the axiom $B_i\varphi \rightarrow D_i\varphi$ that is called *realism* and the constraint

$$\forall t \in \text{Points}_{\mathcal{R}} \exists t' \in \text{Points}_{\mathcal{R}} : tD_it' \text{ and } tB_it'$$

giving rise to the axiom $D_i\varphi \rightarrow \neg B_i\neg\varphi$ that is called *weak realism* [18].

DEFINITION 3 (SEMANTICS).

Let \mathcal{M} be a model with associated set $\text{Points}_{\mathcal{R}}$ and $(r, m) \in \text{Points}_{\mathcal{R}}$. Then the relation \models is defined by:

$$\begin{aligned} \mathcal{M}, r, m &\models p \text{ iff } p \in \pi(r[m]) \\ \mathcal{M}, r, m &\models \neg\varphi \text{ iff } \mathcal{M}, r, m \not\models \varphi \\ \mathcal{M}, r, m &\models \varphi \wedge \psi \text{ iff } \mathcal{M}, r, m \models \varphi \text{ and } \mathcal{M}, r, m \models \psi \\ \mathcal{M}, r, m &\models \bigcirc\varphi \text{ iff } \mathcal{M}, r, m+1 \models \varphi \\ \mathcal{M}, r, m &\models \varphi \mathcal{U} \psi \text{ iff } \exists k \geq m \text{ with } \mathcal{M}, r, k \models \psi \text{ and } \forall l \text{ with } \\ &\quad m \leq l < k \text{ we have that } \mathcal{M}, r, l \models \varphi \\ \mathcal{M}, r, m &\models B_i\varphi \text{ iff } \mathcal{M}, r', m' \models \varphi \text{ for all } (r', m') \text{ with } \\ &\quad (r, m)B_i(r', m') \\ \mathcal{M}, r, m &\models D_i\varphi \text{ iff } \mathcal{M}, r', m' \models \varphi \text{ for all } (r', m') \text{ with } \\ &\quad (r, m)D_i(r', m') \\ \mathcal{M}, r, m &\models I_i\varphi \text{ iff } \mathcal{M}, r', m' \models \varphi \text{ for all } (r', m') \text{ with } \\ &\quad (r, m)I_i(r', m') \end{aligned}$$

*We say a formula φ is satisfiable in \mathcal{M} if there is a run $r \in \mathcal{R}$ and $m \in \mathbb{N}_0$ such that $\mathcal{M}, r, m \models \varphi$, and simply satisfiable if it is satisfiable in some model \mathcal{M} . φ is said to be valid if $\mathcal{M}, r, m \models \varphi$ for all runs $r \in \mathcal{R}_{\mathcal{M}}$ and $m \in \mathbb{N}_0$ in all models \mathcal{M} . We define the logic **BDI_{LTL}** as the set of formulas that are valid on the class of **BDI_{LTL}**-models.*

3.2 Fragments

We are interested in establishing lower complexity bounds for fragments of the logic $\mathbf{BDI}_{\text{LTL}}$ that are still expressive enough to allow reasoning about mental states of other agents. We have reviewed relevant results in Section 2. Based on the summary overview provided there, there are just a few fragments of $\mathbf{BDI}_{\text{LTL}}$ that may be in \mathbf{NP} (or even have a lower complexity). A number of options to lower complexity are not that interesting since we want to be able to reason about the mental states of agents. Therefore, single agent fragments, or single proposition fragments are not interesting. Fragments that may be interesting clearly need to restrict the temporal depth (i.e. nesting of temporal operators), modal depth (i.e. depth of nesting of mental attitude operators), and may need to restrict the number of propositional atoms used.

By inspecting the tables for linear time logic and multi-agent knowledge/belief logics, to obtain fragments that possibly are in \mathbf{NP} the following is clear:

- modal depth, i.e. nesting of B , D , and I operators needs to be bounded, and
- no nesting of temporal operators is allowed at all if unbounded number of propositional atoms are allowed, or temporal depth needs to be bounded.

This means that there are only two fragments of the *language* that we need to consider in the remainder when we are looking for fragments for which satisfiability is in \mathbf{NP} : (1) the fragment with finitely many propositional atoms, limited temporal height, and bounded nesting of other modal operators representing mental attitudes, and (2) the fragment with no nesting of temporal operators and bounded nesting of other modal operators representing mental attitudes. Apart from the language, we also consider slight modifications of the semantics as discussed above as well as the interaction axioms below.

Formally, we use $\text{depth}^m(\varphi)$ to denote the *modal depth* of φ with respect to modal operators B_i , I_i , and D_i . We define $\text{depth}^m(p) = 0$, $\text{depth}^m(\neg\varphi) = \text{depth}^m(\varphi)$, $\text{depth}^m(\bigcirc\varphi) = \text{depth}^m(\varphi)$, $\text{depth}^m(\varphi \circ \psi) = \max\{\text{depth}^m(\varphi), \text{depth}^m(\psi)\}$ for $\circ \in \{\wedge, \vee\}$, and $\text{depth}^m(O_i\varphi) = \text{depth}^m(\varphi) + 1$. Similarly, we define the *temporal depth* $\text{depth}^t(\varphi)$ of φ . In this case the number of nested temporal operators are counted.

4. TABLEAU METHOD FOR BDI LOGIC

We will present a tableau-based method for the satisfiability problem of $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X},\mathbf{Y},\mathbf{Z}}$. We also consider settings where mental attitudes interact such as the conditions of realism and weak realism introduced above. The method has been used by others as well and our approach builds upon the work of [23] which discusses the tableau method for temporal logic extended with either a belief or a knowledge operator. The extension we propose also uses ideas presented in [9, 13, 18]. Although a tableau algorithm for linear time and BDI-operators has been provided in [18] as well, the main concern of our paper, i.e. the complexity of $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X},\mathbf{Y},\mathbf{Z}}$ and associated bounded fragments is not discussed in [18].

4.1 Basic Definitions

We use $\text{sub}(\varphi)$ to denote the *set of subformulas* of φ . Formulas are classified as either α - or β -formula (or none of these). Figure 1 shows which formulas are α - and which formulas are β -formulas. We also refer to α^O -formulas (see Fig-

α	α_1	α_2			
$\neg\neg\varphi$	φ	φ			
$\neg\bigcirc\varphi$	$\bigcirc\neg\varphi$	$\bigcirc\neg\varphi$			
$\varphi \wedge \psi$	φ	ψ			
$\neg(\varphi\mathcal{U}\psi)$	$\neg\psi$	$\neg\varphi \vee \neg\bigcirc(\varphi\mathcal{U}\psi)$			
β	β_1	β_2	α^O	α_1^O	α_2^O
$\neg(\varphi \wedge \psi)$	$\neg\varphi$	$\neg\psi$	$O_i\varphi$	φ	$O_i\varphi$
$\varphi\mathcal{U}\psi$	ψ	$\varphi \wedge \bigcirc(\varphi\mathcal{U}\psi)$			

Figure 1: α , β , α^O -rules with O some modal operator

ure 1) as α -formulas; the reason for introducing this separate class of rules is that they are needed when the accessibility relation associated with O_i is reflexive, a case that is treated differently from other properties. (Note, that some formulas match non of these cases, e.g. $\neg O_i\varphi$.) We note that for each α -formula (resp. β -formula) we have that $\alpha \leftrightarrow \alpha_1 \wedge \alpha_2$ (resp. $\beta \leftrightarrow \beta_1 \vee \beta_2$). A set Σ of formulas is said to be α -closed (resp. β -closed, α^O -closed), if for each α -formula (resp. β -formula, α^O -formula) we have $\{\alpha_1, \alpha_2\} \subseteq \Sigma$ (resp. $\{\beta_1, \beta_2\} \cap \Sigma \neq \emptyset$, $\{\alpha_1^O, \alpha_2^O\} \subseteq \Sigma$). A set Σ of formulas is said to be *fully expanded* if for each $\varphi \in \Sigma$ and for all subformulas $\psi \in \text{sub}(\varphi)$, $\psi \in \Sigma$ or $\neg\psi \in \Sigma$. A set of formulas is called *blatantly inconsistent* (*b-inconsistent* for short) if it contains \perp , $\neg\top$, or two complementary pairs of formulas $\varphi, \neg\varphi$. If a set is not b-inconsistent it is *b-consistent*.

In the remainder of this paper, we use O to refer to arbitrary BDI operators, i.e. $O \in \{B, D, I\}$, and \mathcal{O} to refer to the corresponding accessibility relation. We say that an operator O is a **T-operator** if the associated accessibility relation \mathcal{O} is reflexive. Similarly, we say that O is a **KD-operator** if the relation is serial, etc. We also write $\mathbf{X} \in \text{schema}(O)$ if O is an **X-operator**, $\mathbf{X} \in L$ and so on.

DEFINITION 4 (PC-TABLEAU). *We call a set Σ of formulas a PC-tableau if Σ is b-consistent, α - and β -closed, and fully expanded. Moreover, we assume that it is α^O -closed if O is a **T-operator**.*

A PC-tableau derived from Σ' is a PC-tableau Σ such that $\Sigma' \subseteq \Sigma$. The set of all PC-tableaux derived from Σ' is denoted $\text{PC}(\Sigma')$. (Note that if Σ' is b-inconsistent then $\text{PC}(\Sigma') = \emptyset$.)

As noted, the use of the rule α^O (see Figure 1) differs from others in that it is applied to all formulas $O\varphi$ iff O is a **T-operator**; it is therefore easier to incorporate this rule into the definition of a PC-tableau. Note that for reflexive relations \mathcal{O} we actually have $\alpha^O \leftrightarrow \alpha_1^O \wedge \alpha_2^O$.

4.2 Tableau Construction

We now discuss the tableau-based method for showing satisfiability of the family of logics $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X},\mathbf{Y},\mathbf{Z}}$. Each variant within this family of logics requires that some modifications are applied to the general and generic algorithm below. We begin with explaining the idea for the particular logic $\mathbf{BDI}_{\text{LTL}}^{\text{KD},\text{KB},\text{KD}}$, and thereafter discuss the modifications that are needed for the other members of our family.

$\mathbf{BDI}_{\text{LTL}}^{\text{KD},\text{KB},\text{KD}}$ -pseudo-structure algorithm.

The tableau approach is based on the idea that tableaux can be used to construct models. Pseudo-structures have the important property that they can be extended to a model if

The algorithm specifies $S = (\hat{Q}, \hat{\rightarrow}, \{\hat{B}_i\}_{i \in \text{Agt}}, \{\hat{D}_i\}_{i \in \text{Agt}}, \{\hat{I}_i\}_{i \in \text{Agt}}, \hat{\pi})$ on input φ_0 .

1. **(Initialisation)** For each $\Delta \in PC(\{\varphi_0\})$ add a state q with $\hat{\pi}(q) = \Delta$ to \hat{Q} .
2. **Repeat until convergence:**
 - (a) *(Modal transitions)* For any $q \in \hat{Q}$, if $\neg O_i \psi \in \hat{\pi}(q)$ set

$$(\star) \quad \Sigma = \{\neg \psi\} \cup \{\chi \mid O_i \chi \in \hat{\pi}(q)\} \cup \text{create_label}_{\text{schema}(O)}^i(\hat{\pi}(q)).$$
 If such a formula does not exist then define

$$\Sigma = \{\chi \mid O_i \chi \in \hat{\pi}(q)\} \cup \text{create_label}_{\text{schema}(O)}^i(\hat{\pi}(q)).$$
 For each $\Delta \in PC(\Sigma)$ if there is a state $q' \in \hat{Q}$ with $\hat{\pi}(q') = \Delta$ add the relation $q \hat{O}_i q'$. Otherwise,

$$(\star\star) \quad \text{create a node } q' \text{ with } \hat{\pi}(q') = \Delta \text{ and add relation } q \hat{O}_i q'.$$
 - (b) *(Temporal transitions)* For any $q \in \hat{Q}$, if $\bigcirc \psi \in \hat{\pi}(q)$ then for each $\Delta \in PC(\hat{\pi}(q)/\bigcirc)$ if there is a state $q' \in \hat{Q}$ with $\hat{\pi}(q') = \Delta$ add the transition $q \hat{\rightarrow} q'$ else add a new state q' with $\hat{\pi}(q') = \Delta$ and add the transition $q \hat{\rightarrow} q'$.
 3. **(Deletion)** Delete a state $q \in \hat{Q}$ if one of the following conditions applies:
 - (a) $\exists \psi \in \hat{\pi}(q)$ such that ψ is (S, q) -temporally inconsistent.
 - (b) $\exists \psi \in \hat{\pi}(q)$ such that $\psi = \bigcirc \chi$ and there is no q' with $q \hat{\rightarrow} q'$.
 - (c) $\exists \psi \in \hat{\pi}(q)$ such that $\psi = \neg O_i \chi$ and there is no q' with $q \hat{O}_i q'$ and $\neg \chi \in \hat{\pi}(q')$.
 - (d) $\exists \psi \in \hat{\pi}(q)$ such that $\psi = O_i \chi$ and there is no q' with $q \hat{O}_i q'$ and $\chi \in \hat{\pi}(q')$ (only if O is a **D**-operator).

Figure 2: $\text{BDI}_{\text{LTL}}^{\text{X,Y,Z}}$ -pseudo-structure algorithm for serial accessibility relations and input φ_0 . We assume that $O \in \{B_i, D_i, I_i\}$ and that \hat{O} denotes the corresponding accessibility relation.

The procedure returns the following set:

1. \emptyset if $\text{X} \in \{\mathbf{K}, \mathbf{KD}\}$.
2. $\{O_i \psi \mid O_i \psi \in \Sigma\} \cup \{\neg O_i \psi \mid \neg O_i \psi \in \Sigma\}$ if $X \in \{\mathbf{KD45}, \mathbf{S5}\}$.
3. $\{O_i \psi \mid O_i \psi \in \Sigma\}$ if $X = \mathbf{S4}$.

Figure 3: $\text{create_label}_{\chi}^i(\Sigma)$ procedure.

and only if the input formula is satisfiable. Accordingly, we present an algorithm that generates pseudo-structures.

DEFINITION 5. A pseudo-structure is a tuple $S = (\hat{Q}, \hat{\rightarrow}, \{\hat{B}_i\}_{i \in \text{Agt}}, \{\hat{D}_i\}_{i \in \text{Agt}}, \{\hat{I}_i\}_{i \in \text{Agt}}, \hat{\pi})$ where \hat{Q} is a (possibly empty) set of states, and $\hat{\rightarrow}, \hat{B}_i, \hat{D}_i,$ and \hat{I}_i are binary relations between states, and $\hat{\pi} : \hat{Q} \rightarrow \mathcal{P}(\mathcal{L})$ assigns sets of formulae to states.

The basic algorithm is called $\text{BDI}_{\text{LTL}}^{\mathbf{KD}, \mathbf{KD}, \mathbf{KD}}$ -pseudo-structure algorithm and is presented in Figure 2. If the algorithm returns a pseudo-structure and the input formula is satisfiable, a BDI_{LTL} -model can be extracted from the structure that witnesses the truth of the formula. In Theorem 2 it is shown that the pseudo-structure contains a state q whose label contains φ if, and only if, the pseudo-structure can be extended to a $\text{BDI}_{\text{LTL}}^{\mathbf{KD}, \mathbf{KD}, \mathbf{KD}}$ -model satisfying φ .

The first step in the algorithm generates nodes each labeled with a PC-tableau derived from $\{\varphi\}$. Then, steps (2a) and (2b) are performed until none of these cases can be applied anymore. Step (2a) generates \hat{O}_i transitions. Depending on the properties of \hat{O}_i (referred to by procedure $\text{schema}(O)$) the procedure $\text{create_label}_{\text{schema}(O)}^i(\Sigma)$ which is

shown in Figure 3 creates the label of a (possibly new) node. Different logics require a different procedure. $\hat{\pi}(q)/\bigcirc$ is defined as $\{\psi \mid \bigcirc \psi \in \hat{\pi}(q)\}$ and $\hat{\pi}(q)/O_i$ analogously.

Condition $(\star\star)$ corresponds to the seriality condition of \hat{O} . In (2b) temporal successors are created. The idea is that if the current state contains a formula $\bigcirc \varphi$ there has to be a \rightarrow -related state satisfying φ . Finally, in step (3) the algorithm deletes states which are not consistent – one way or another. Of particular interest is step (3a) which removes nodes in which an eventuality formula cannot be satisfied anymore. This step involves the notion of temporally consistent formulas, which is defined next.

DEFINITION 6 (TEMPORALLY CONSISTENT). Given a pseudo-structure S and a state $q \in \hat{Q}_S$ a formula φ is said to be (S, q) -temporally consistent iff if $\varphi = \psi_1 \mathcal{U} \psi_2$ then there is a state q' reachable from q via the transitive closure of $\hat{\rightarrow}$ such that $\psi_2 \in \hat{\pi}(q')$. If φ is not (S, q) -temporally consistent it is said to be (S, q) -temporally inconsistent.

In step 3(d) states are deleted that are not consistent with the fact that an operator O is a **KD**-operator, which requires a successor state in the corresponding models.

Modifications: $\text{BDI}_{\text{LTL}}^{\text{X,Y,Z}}$ -pseudo-structure algorithms.

Before we state our general result, we briefly consider what needs to be modified to cater for other logics than $\text{BDI}_{\text{LTL}}^{\mathbf{KD}, \mathbf{KD}, \mathbf{KD}}$. We consider the general case and define a generic $\text{BDI}_{\text{LTL}}^{\text{X,Y,Z}}$ -pseudo-structure algorithm as follows. First, for operators and associated relations O that have other properties than seriality the labeling of nodes in (\star) needs to be modified. The required modifications are listed in Figure 3 for the different cases that we consider here.

Finally, for operators that are not **D**-operators, the cases that relate to the seriality of the accessibility relation in the algorithm need to be disregarded. This applies in particular to condition $(\star\star)$ which should be ignored when dealing with such operators. Finally, also the deletion of states needs to be modified and only the cases 3(a)-3(c) should be executed while case 3(d) in Figure 2 needs to be ignored.

4.3 Soundness and Completeness

We now consider the soundness and completeness of the algorithm. The proofs of the results are fairly standard (cf. [23, 9, 13, 18]) and we focus on some of the basic ideas.

A pseudo-structure $S = (\hat{Q}, \hat{\rightarrow}, \{\hat{B}_i\}_{i \in \text{Agt}}, \{\hat{D}_i\}_{i \in \text{Agt}}, \{\hat{I}_i\}_{i \in \text{Agt}}, \hat{\pi})$ is said to be a $\text{BDI}_{\text{LTL}}^{\text{X,Y,Z}}$ -tableau for φ if the following 8 conditions are satisfied: (1) There is a state $q \in \hat{Q}$ such that $\varphi \in \hat{\pi}(q)$. (2) For each $q \in \hat{Q}$, $\hat{\pi}(q) \in PC(\hat{\pi}(q))$. (3) If $\mathbf{T} \in \text{schema}(O)$ then if $O_i \psi \in \hat{\pi}(q)$ then $\psi \in \hat{\pi}(q)$. (4) If $\mathbf{D} \in \text{schema}(O)$ then if $O_i \psi \in \hat{\pi}(q)$ then there is a state q' with $q \hat{O}_i q'$. (5) If $\mathbf{4} \in \text{schema}(O)$ then if $O_i \psi \in \hat{\pi}(q)$ then for all q' with $q \hat{O}_i q'$, $O_i \psi \in \hat{\pi}(q')$. (6) If $\mathbf{5} \in \text{schema}(O)$ then if $q \hat{O}_i q'$ and $q \hat{O}_i q''$ and $O_i \psi \in \hat{\pi}(q')$ then $\{O_i \psi, \psi\} \subseteq \hat{\pi}(q'')$. (7) If $\bigcirc \psi \in \hat{\pi}(q)$ then there is a $q' \in \hat{Q}$ such that $q \hat{\rightarrow} q'$. (8) If $\bigcirc \psi \in \hat{\pi}(q)$ then for all $q' \in \hat{Q}$ with $q \hat{\rightarrow} q'$ it holds that $\psi \in \hat{\pi}(q')$.

Conditions 3, 4, 5, and 6 correspond to reflexivity, seriality, transitivity, and Euclideanity, respectively. The following theorems are more or less standard.

THEOREM 1 (TABLEAU SATISFIABILITY). A formula φ is $\text{BDI}_{\text{LTL}}^{\text{X,Y,Z}}$ -satisfiable iff there is a $\text{BDI}_{\text{LTL}}^{\text{X,Y,Z}}$ -tableau for φ .

PROOF SKETCH. \Rightarrow : It is easily seen that one can define a tableau from a satisfying model. \Leftarrow : The set of timelines \mathcal{R} is obtained by unravelling the relation \rightarrow , enforcing the satisfaction of eventuality formulae (cf. [23] for details). The definition of the state-based $\hat{\mathcal{O}}$ -relations to point-based \mathcal{O} accessibility relations is also done according to [23]: For example we set $(r, m)\mathcal{O}_i(r', m')$ if $r(m)\hat{\mathcal{O}}_i r'(m')$. Minor modifications are necessary to ensure that \mathcal{O} is a *schema*(\mathcal{O})-operator (e.g. taking the reflexive transitive closure, etc.). \square

THEOREM 2 (SOUND-, COMPLETENESS). *The algorithm terminates on all inputs and φ is $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X}, \mathbf{Y}, \mathbf{Z}}$ -satisfiable iff the $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X}, \mathbf{Y}, \mathbf{Z}}$ -algorithm on input φ returns a $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X}, \mathbf{Y}, \mathbf{Z}}$ -tableau for φ iff the structure returned by the algorithm contains a state q containing φ .*

PROOF SKETCH. Termination of the algorithm is guaranteed as there are only finitely many different PC-tableaux and the algorithm does not create nodes twice. Soundness is proved following similar steps as in [23]: It is shown that if the algorithm returns a pseudo-structure for φ then it is actually a tableau for φ (this is achieved by verifying the 8 conditions of a tableau). For the completeness one shows that if there is no state q in the returned structure, then φ is not satisfiable (cf. [23, Th.4]). The modal operators are treated independently. \square

4.4 Interaction Axioms

Finally, we consider the interaction axioms. The realism axiom is given by $B_i\varphi \rightarrow D_i\varphi$. In order to extend our tableau method an additional alpha rule is introduced: $\alpha = B_i\varphi$ and $\alpha_1 = B_i\varphi$ and $\alpha_2 = D_i\varphi$. The condition that we need to add to cover for this axiom is the requirement that a PC-tableau is also closed under this new rule (see Df. 4).

Similarly, for the weak realism axiom $D_i\varphi \rightarrow \neg B_i\neg\varphi$, a second new α -rule is introduced: $\alpha = D_i\varphi$ and $\alpha_1 = D_i\varphi$ and $\alpha_2 = \neg B_i\neg\varphi$, and the definition of a PC-tableau needs to be modified accordingly.

It is not difficult to see that the sound- and completeness results from Section 4.3 also hold with these extensions. The additional α -rules for realism and weak realism give rise to the corresponding rules

9. If $B_i\varphi \in \hat{\pi}(q)$ then $D_i\varphi \in \hat{\pi}(q)$; and
10. if $D_i\varphi \in \hat{\pi}(q)$ then $\neg B_i\neg\varphi \in \hat{\pi}(q)$, respectively,

in the $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X}, \mathbf{Y}, \mathbf{Z}}$ -tableau (see also [18, 23]). The corresponding model is constructed from such a table in the same way as in the cases without interaction (cf. Theorem 1) and the completeness proof of Theorem 2 is done analogously (see e.g. [18, 9]).

5. COMPLEXITY OF SATISFIABILITY

In this section we consider the complexity of the $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X}, \mathbf{Y}, \mathbf{Z}}$ -satisfiability problem. In [13] a tableau-based decision procedure has been used to prove **PSPACE** membership of the multi-agent logics **K**, **T**, **S4**, **S5**, and **KD45**. In [9] a **PSPACE** tableau algorithm for a **BDI** logic combining **S4**, **K**, and **KD** operators has been presented and in [18, 23] tableau-based algorithms for linear-time and combinations of other modalities were given. However, the complexity of the latter algorithms has not been analysed.

“Standard” **LTL**-tableau constructions have been shown implementable in **PSPACE** [22]. The algorithm presented here when executed with purely temporal formulae is essentially equivalent to that of [22]. The next result shows that the addition of other modal operators does not increase the complexity.

THEOREM 3. *The $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X}, \mathbf{Y}, \mathbf{Z}}$ -satisfiability problem is **PSPACE**-complete for all $\mathbf{X}, \mathbf{Y}, \mathbf{Z} \in L$.*

PROOF SKETCH. The lower bound follows from **LTL**-satisfiability, cf. e.g. [5]. We sketch the upper bound. In the following we take $d = \text{depth}^m(\varphi)^{\mathcal{O}(1)}$.

(I) Purely modal part (step 2(a) in our algorithm): In [13] it was shown that the length of sequences of subsequent states generated by the tableau algorithm is bounded by d . (Note, that we abstracted from the calculation of PC-tableaux. The treatment is standard.) The “tree” consisting of all these polynomial length sequences is searched in a depth first search manner using only polynomial space. This procedure generalizes to multiple modal operators.

(II) Purely temporal part (step 2(b) in our algorithm): In [22] a **PSPACE**-tableau algorithm for **LTL** is presented. The main observation is that if an **LTL** formula is satisfiable then it is satisfiable on a path $q_0q_1 \dots q_m$ such that for some q_j , $j \leq m$ with $q_m \rightarrow q_j$ and $m \leq 2^{\mathcal{O}(|\varphi|)}$. The following polynomial space algorithm implements this idea: Guess state q_j and guess valid subsequent successor states q (for at most $2^{\mathcal{O}(|\varphi|)}$ steps) until $q = q_j$ for some state in which all eventualities are fulfilled (φ is satisfiable). In memory, only the counter, the current state and q_j are kept.

(III) We consider $\mathbf{BDI}_{\text{LTL}}$. As noted in [10] time and epistemic operators are independent from each other which allows for a combination of (I) and (II). From (I) we know that the number of consecutive epistemic steps is bounded by d . Now, each time we are in step 2(a) we apply the depth first search strategy from (I) and each time we execute step 2(b) we try to build the infinite trace as in (II). The number of “temporal traces” is bounded by $|\varphi| \cdot d$. Hence, one has to store at most $|\varphi| \cdot d$ counters, current states, and the states guessed to indicate the entry point of a loop for the temporal part and $|\varphi| \cdot d$ states constituting the current “epistemic path” which is kept in memory (possibly interrupted by a temporal path). (In addition to some other book keeping operations needed for the depth first search.) \square

In a non-temporal setting, it was shown that adding various interaction axioms does not increase the complexity [9]. This is also the case in the temporal setting considered here. The result follows immediately since the interaction axioms do only require the application of some additional α -rules as explained above.

COROLLARY 4. *The $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X}, \mathbf{Y}, \mathbf{Z}}$ -satisfiability problem assuming realism or weak realism is **PSPACE**-complete.*

5.1 Bounded Temporal and Modal Depth

Here we consider fragments of $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X}, \mathbf{Y}, \mathbf{Z}}$ with bounds on the temporal depth $\text{depth}^t(\varphi)$ and modal depth $\text{depth}^m(\varphi)$ of a formula φ .

To be precise, we define the fragments of $\mathbf{BDI}_{\text{LTL}}^{\mathbf{X}, \mathbf{Y}, \mathbf{Z}}$ considered here using the notions of temporal and modal depth. We define $\mathcal{L}^{i,j}$ as the set of formulas $\varphi \in \mathcal{L}$ with $\text{depth}^t(\varphi) \leq i$ and $\text{depth}^m(\varphi) \leq j$.

The upper bound of the next result follows from Theorem 3 and the lower bound from [5] where the satisfiability problem for **LTL** with a temporal bound of ≥ 2 is shown to be **PSPACE**-complete.

PROPOSITION 5 ([5]). *Let $i, j \in \mathbb{N}$ and $i \geq 2$. The $\mathbf{BDI}_{\text{LTL}}^{X,Y,Z}$ -satisfiability problem over the language $\mathcal{L}^{i,j}$ is **PSPACE**-complete. The same result holds over the class of models that satisfy (weak) realism.*

The previous result is negative. However, the complexity improves if the temporal depth is at most 1.

THEOREM 6. *Let $i \in \mathbb{N}$. The $\mathbf{BDI}_{\text{LTL}}^{X,Y,Z}$ -satisfiability problem over the language $\mathcal{L}^{1,i}$ is **NP**-complete. The same result holds over the class of models that satisfy (weak) realism.*

PROOF SKETCH. In [5] it is shown that if an **LTL** formula with $\text{depth}^t(\varphi) \leq 1$ is satisfiable then satisfiability can be witnessed by an initial polynomial length prefix of a path. This prefix can be non-deterministically guessed and verified in polynomial time. Moreover, as shown in [13, 9] the purely modal tableaux have $|\varphi|^{\mathcal{O}(i)}$ many nodes. Combining both results and inspecting the proof sketch of Theorem 3 shows that for the full logic the number of nodes in the tableau is also bounded by $|\varphi|^{\mathcal{O}(i)}$. (Here, it is important to note that there can be at most one alternation between epistemic and modal transitions along each path in the tableau.) Thus, we can guess a tableau of polynomial size and check whether it satisfies φ . \square

Finally, we consider the case in which the temporal and modal depth are bounded and additionally only finitely many propositional symbols are available. We note that a finite set of propositions is often of less practical interest (e.g. settings requiring natural numbers usually require an unbounded number of propositional symbols). For the purely temporal fragment it has been proven that the problem can be checked in logarithmic deterministic space [5]. In [12], on the other hand, satisfiability of the purely modal fragment has been shown to be solvable in linear deterministic time. The proof of [5], however, can directly be used to show that $\mathbf{BDI}_{\text{LTL}}^{X,Y,Z}$ -satisfiability for formulae of $\mathcal{L}^{i,j}$, i, j fixed, can be checked in logarithmic deterministic space. The basic idea relies on the observation that there are only finitely many inequivalent formulae over $\mathcal{L}^{i,j}$. For each class of equivalent formulae we identify a ‘‘canonical formula’’. Then, given a formula φ of which the satisfiability should be checked simple rewrite rules can be applied to determine the canonical representation ψ of φ . If ψ does not correspond to the canonical representation of \perp it is satisfiable.

THEOREM 7. *Let $i, j \in \mathbb{N}$. The $\mathbf{BDI}_{\text{LTL}}^{X,Y,Z}$ -satisfiability problem over $\mathcal{L}^{i,j}$ over a finite set of propositional atoms can be solved in deterministic logarithmic space. This is also true for the class of models that satisfy (weak) realism.*

6. REASONING ABOUT MENTAL STATES

We have studied various fragments of the linear time BDI logic $\mathbf{BDI}_{\text{LTL}}$ and obtained results on the complexity of the corresponding satisfiability problems. Our motivation for examining these fragments has been that agents need to be able to reason about other agents’ mental states and logic seems one of the most suitable tools to do so. However,

an agent also needs to be able to do so within reasonable amounts of space and time. We have shown that fragments exist of which the complexity can be reduced to **NP**. As argued above, as a first step towards a logic that can actually be used, these results are promising. Here we briefly consider informally whether these fragments are also satisfactory for the main task we had in mind, i.e. for the representation and reasoning with mental states of other agents.

In order to evaluate this, we briefly introduce and discuss some *minimal criteria*. In order to support reasoning about other agents’ mental states, we argue that the minimal requirement a logic needs is that the logic (i) is able to discriminate between *informational attitudes* such as beliefs and knowledge and *motivational attitudes* such as desires, goals, and intentions; (ii) facilitates reasoning about a *finite number of nestings* of mental attitudes; (iii) facilitates reasoning about *any finite* number of agents; and (iv) is able to discriminate between types of motivational states, and, ideally would support reasoning about the class of *deadline goals* which subsumes *achievement* and *maintenance* goals.

Criterion (i) has motivated us to study fragments of $\mathbf{BDI}_{\text{LTL}}$. This is one of the more well-known types of agent logics and clearly distinguishes between informational and motivational states as parts of an agent’s state.

Criterion (ii) is in part motivated by results from cognitive science which inform us that a limited depth of mental operators seems sufficient for reasoning about mental states as humans are not able to nest such operators to a depth of more than 3 [7]. On the other hand, in the context of agent communication and reasoning about speech acts, one finds that one quickly needs at least three levels of nesting (see e.g. the example of a decision rule to inform another agent below). Given these basic results about depth of nesting of mental attitude operators, it seems reasonable to bound the depth of such operators. The $\mathbf{BDI}_{\text{LTL}}$ fragments that we showed to be in **NP** clearly support such limited nesting.

Criterion (iii) is an obvious criterion given that we are motivated by logics that allow agents to reason about other agents’ mental state. This motivated us to introduce the multi-agent variant of BDI logic. From a semantic and a complexity perspective this requirement poses no problems.

Criterion (iv) most clearly distinguishes our work from that of others, in particular [9], which studies a multi-agent logic called **TEAMLOG** that does not incorporate time. Being able to support some form of reasoning about time, however, greatly increases the expressivity and in particular allows to distinguish between various kinds of goals. For example, using a single \mathcal{U} operator we are already able to represent *deadline goals*. Moreover, it is clear that achievement goals can be represented by $\diamond\phi$ and maintenance goals by $\square\phi$. The fragments we showed to be in **NP** support making these distinctions. However, only the fragment with finitely many atoms supports nesting of temporal operators (Th. 7). The fragment that allows for infinitely many atoms (Th. 6) thus does not allow reasoning about e.g. persistence goals $\diamond\square\varphi$. It would be interesting to investigate if that fragment could be extended with such specific combinations (while not allowing nesting of arbitrary temporal operators).

Clearly, group attitudes such as common knowledge and common or joint intentions cannot be defined using the bounded fragments we discussed. Introducing these concepts immediately blows up complexity (e.g. [9, 13]). Of course, we can define more basic notions such as ‘‘everybody in a group

believes φ ", i.e. $E-B_G(\varphi) \leftrightarrow \bigwedge_{i \in G} B_i\varphi$. Even without such stronger notions, however, agents can coordinate their behavior. The idea would be that agents can incorporate reasoning based on a **BDI_{LTL}** fragment into their decision making algorithms. This would already allow for action choices based on decision rules of the form: (i) **if** $B_i I_j \varphi$ **then do**(a), (ii) **if** $B_i B_j \varphi$ **then do**(a), and (iii) **if** $B_i D_j (K_j \varphi \vee K_j \neg \varphi) \wedge B_i \varphi$ **then inform**(φ).

7. CONCLUSION AND FUTURE WORK

We have discussed the issue of reasoning about mental states of other agents and argued that the BDI logic **BDI_{LTL}** offers a suitable tool to do so. We have studied several fragments of **BDI_{LTL}** and showed complexity of the satisfiability problem for these fragments to be in **NP**. We also introduced a very generic tableau method to do so.

From a complexity point of view there are many ways to continue the search for practical and useful fragments of **BDI_{LTL}**. For example, in [6] it is shown that the satisfiability problem for a special semantics for temporal logic with knowledge called *XL5* is **NP**-hard and it is interesting to study whether similar techniques can be applied to **BDI_{LTL}**. [16] discusses the complexity of the satisfiability problem for a range of multi-modal logics *restricted to the Horn fragment* including e.g. **KD_n**, **KD45_n**, etc. The restriction to the Horn fragment is relevant because it may provide practical extensions to e.g. Prolog. [16] also investigates bounded modal depth but does not discuss fragments that include both informational and motivational operators and does not discuss temporal operators either. Another line of research would involve looking at **CTL** instead of **LTL**, and identify complexity classes for fragments of **BDICTL**.

In the future, we would also like to consider agents that have *perfect recall* or do not *learn* and extend our analysis in this respect. Both properties are of practical importance. However, usually reasoning becomes computationally much harder in the presence of these properties.

Finally, it is interesting to experiment in practice with implemented reasoners to determine what is feasible. Moreover, if we want to incorporate a restricted BDI logic into the decision making component of a software agent, we not only need a reasoner, but we will also need ways to efficiently update sets of **BDI_{LTL}** formulas when the agent receives new information from its environment or through communication with other agents.

8. REFERENCES

- [1] P. Blackburn, M. de Rijke, and Y. Venema. *Modal Logic*. Cambridge University Press, 2001.
- [2] R. H. Bordini, M. M. Dastani, J. Dix, and A. E. F. Seghrouchni. *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, 2005.
- [3] R. H. Bordini, M. M. Dastani, J. Dix, and A. E. F. Seghrouchni. *Multi-Agent Programming: Languages, Tools and Applications*. Springer, 2009.
- [4] P. R. Cohen, H. R. Levesque, and I. Smith. On team formation. In J. Hintikka and R. Tuomela, editors, *Contemporary Action Theory. Synthese*. 1997.
- [5] S. Demri and P. Schnoebelen. The complexity of propositional linear temporal logics in simple cases. Technical report, Lab. Spécification et Vérification, CNRS and ENS de Cachan, France, 1997.
- [6] C. Dixon, M. Fisher, and B. Konev. Taming the complexity of temporal epistemic reasoning. In *Proceedings of the Seventh International Symposium on Frontiers of Combining Systems (FroCoS)*, 2009.
- [7] B. Dunin-Keplicz and R. Verbrugge. Awareness as a vital ingredient of teamwork. In *Proceedings of the 5th international joint conference on Autonomous agents and multiagent systems*, pages 1017–1024, 2006.
- [8] E. H. Durfee. Scaling up agent coordination strategies. *Computer*, 34(7):39–46, 2001.
- [9] M. Dziubinski, R. Verbrugge, and B. Dunin-Keplicz. Complexity issues in multiagent logics. *Fundamenta Informaticae*, 75(1-4):239–262, 2007.
- [10] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning about Knowledge*. MIT Press, 1995.
- [11] M. Fisher. Implementing BDI-like Systems by Direct Execution. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, pages 316–321, 1997.
- [12] J. Halpern. The effect of bounding the number of primitive propositions and the depth of nesting on the complexity of modal logic. *Artificial Intelligence*, 75(2):361–372, 1995.
- [13] J. Halpern and Y. Moses. A guide to completeness and complexity for modal logics of knowledge and belief. *Artificial Intelligence*, 54:319–379, 1992.
- [14] J. Halpern and M. Vardi. The complexity of reasoning about knowledge and time. i. lower bounds. *Journal of Computer System Sciences*, 38(1):195–237, 1989.
- [15] W. G. Kennedy, M. D. Bugajska, A. M. Harrison, and J. G. Trafton. “like-me” simulation as an effective and cognitively plausible basis for social robotics. *International Journal of Social Robotics*, 1(2):181–194, 2009.
- [16] L. A. Nguyen. On the complexity of fragments of modal logics. In *Advances in Modal Logic*, pages 249–268, 2005.
- [17] J. Nick R. Commitments and conventions: The foundation of coordination in multi-agent systems. *The knowledge engineering review*, 8:223–250, 1993.
- [18] A. S. Rao. Decision procedures for propositional linear-time belief-desire-intention logics. In *Intelligent Agents II: Agent Theories, Architectures, and Languages*, pages 33–48, 1996.
- [19] R. A. Schmidt and D. Tishkovsky. A Decidable Dynamic Logic for Agents with Motivational Attitudes. In *Proceedings of the Workshop on Methods for Modalities-2 (M4M’01)*, 2001.
- [20] M. Tambe. Agent architectures for flexible, practical teamwork. In *Proceedings of the National Conference on Artificial Intelligence*, pages 22–28, 1997.
- [21] W. van der Hoek and M. Wooldridge. Towards a logic of rational agency. *Logic Journal of IGPL*, 11(2):135–159, 2003.
- [22] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56:72–99, 1983.
- [23] M. Wooldridge, C. Dixon, and M. Fisher. A tableau-based proof method for temporal logics of knowledge and belief. *Journal of Applied Non-Classical Logics*, 8:225–258, 1998.

Agent-Based System Development I

Scenarios for System Requirements Traceability and Testing

John Thangarajah
RMIT University
Melbourne, Australia
johnt@rmit.edu.au

Gaya Jayatilleke
RMIT University
Melbourne, Australia
gaya.jayatilleke@rmit.edu.au

Lin Padgham
RMIT University
Melbourne, Australia
lin.padgham@rmit.edu.au

ABSTRACT

Scenarios in current design methodologies, provide a natural way for the users to identify the inputs and outputs of the system revolving around a particular interaction process. A scenario typically consists of a sequence of steps which captures a particular run of the system and satisfies some aspect of the requirements. In this work we add additional structure to the scenarios used in the Prometheus agent development methodology. This additional structure then facilitates both traceability and automated testing. We describe our process for mapping the scenarios and their steps to the initial detailed design, where we then maintain the traceability as the design develops. The structured action lists that we define for both scenarios and their variations provides the basis for facilitating automated testing of system behavior. We describe how we use the newly defined structure within the scenarios to facilitate testing, describing how we automate test case generation, execution and analysis.

Categories and Subject Descriptors

D.2.5 [Software]: Software Engineering

General Terms

Design, Reliability, Verification

Keywords

Agent Oriented Software Engineering, Requirements Testing

1. INTRODUCTION

Agent Oriented Software Engineering (and the related agent development platforms or languages) is an approach to building complex systems which has been shown to be very efficient (on average 350% faster development time) compared to standard Java development [1]. Many of the agent system development methodologies (e.g. Tropos [4], Prometheus [8], Roadmap [9]) use some kind of scenario or use case development in the initial stages of agent system specification and design as a way of exploring and developing the specification of a software system.

Cite as: Scenarios for System Requirements Traceability and Testing, John Thangarajah, Gaya Jayatilleke and Lin Padgham, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 285-292.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

In this work we extend the scenarios used in the Prometheus methodology to provide a more precise and better structured scenario specification, which can then be used for the important aspects of traceability and requirements testing.

Scenarios provide a natural way for end users or clients to understand the system that is being specified/designed. As such they are a very useful tool for fleshing out what is expected of a system. By making them more precise and structured they can also play an important role in understanding the implementation, and ensuring that things specified in scenarios are in fact implemented. They are then also a natural candidate for a model based approach to requirements testing.

The scenarios used within Prometheus are already more structured than standard object-oriented use cases, in that the steps of a scenario are specified objects within the design: percepts, actions, goals and sub-scenarios. This allows reasoning about the relationship of scenarios to other design entities - for example it is possible to ensure that the set of scenarios “covers” the system goals.¹ In this work we show how the goals within scenarios (and their sub-scenarios) can be mapped down to structures of events and plans that are then mapped to code. This provides traceability from specification, through code, to design.

The Prometheus scenarios allow for specification of alternatives to the sequence described. However, the alternatives are described only in natural language. It would of course be possible to develop alternatives as fully specified scenarios. However there is a balance between specifying sufficient for a good understanding of the system, versus burying the reader in unnecessary detail which can make it harder to understand the system essentials. In this work we slightly extend the notion of scenario alternatives, to require the specification of actions that would be generated from each alternative. This then provides us with a model which we can use as a basis for requirements testing. Essentially we can test the system by providing the trigger for each scenario (in various situations), and then ensuring that the actions generated are consistent with either the main scenario, or one of the alternatives specified.

In the following sections we first describe the existing representation of scenarios in Prometheus, and then describe our extensions, including the consistency checking that these allow us to accomplish. Secondly we show how we use sce-

¹Coverage of the system goals by a scenario involves a cut through the goal hierarchy so that each goal is “covered” either by having an ancestor that is in the cut, or by having all of its children in the cut.

narios (in particular the goals within scenario steps) to generate skeleton designs which we then annotate in order to maintain traceability back to the scenario descriptors. Finally we describe how these scenario specifications are used to provide the oracle for testing when scenario triggers are inserted into the system that has been initialized to different configurations. We then conclude with a summary of what has been accomplished, and a brief discussion of the future directions for this work.

2. SCENARIO SPECIFICATION

In this section we provide a specification for scenarios, by extending the current representation in the Prometheus methodology.

The Prometheus methodology supports the complete development of agent systems from specification, design, implementation and testing/debugging [8]. It consists of three key phases that guide the developer with a well defined set of artifacts and process: *System specification* is where the system interface is specified in terms of inputs (percepts), outputs (actions), the actors, scenarios (akin to use cases in traditional Object-Oriented design) and the functionality is identified via goals and roles of the system; *Architectural design* where the internals of the system are specified in terms of agents and communication protocols between them; and *detailed design* where each agents internals are detailed to a level that can be readily implemented in a BDI based agent platforms such as JACK^{TM2}. The methodology is supported by the Prometheus Design Tool (PDT) [7].

A scenario captures a particular run of the system that typically covers a subset of the requirements. In other words, scenarios are a way of describing and extending the understanding of the requirements at the specification phase. In Prometheus a scenario is defined as a sequence of steps that is initiated by a trigger such as a percept, an internal event of the agent or time. Possible steps in a scenario are achieving a *goal*, performing an *action*³, receiving a *percept*, performing another (*sub*) *scenario* or *other* step types not covered by the above (for example, awaiting a response). For example, in building an on-line bookstore application, two possible scenarios that need to be covered are, a user placing a new book order and querying about a delay in the delivery of an order. Prometheus also allows for a textual description of scenario variations. For example in the book ordering scenario shown in figure 1 there may be a scenario variation described, stating that if there is insufficient stock, then the user is notified that the book is out of stock and the system will engage in a process to order more stock.

2.1 Extended Scenario Specification

At present these scenarios are used only for requirements elicitation and not as a basis for generating system tests. We extend the current definition of scenarios with a structure and detail that allows us to propagate the scenario information to the implementation constructs through the detailed design process of Prometheus and use them in generating scenario based system test cases. Using the new structure of the scenario, we are also able to assist the developer in

²JACK is the commercial platform developed by Agent Oriented Software www.aosgrp.com.

³An action may map to one or more implementation constructs.

Type	Name	Description
<input type="checkbox"/> Percept	BookOrderPercept	Contains cus
<input type="checkbox"/> Goal	CheckStock	Check if the
<input type="checkbox"/> Action	ConfirmStock	Confirm have
<input type="checkbox"/> Scenario	GetPayment Scenario	get&confirm
<input type="checkbox"/> Goal	ArrangeDelivery	Arrange proc
<input type="checkbox"/> Goal	UpdateStock	Update stock

Figure 1: Example: Book Order Scenario Steps

Type	Name	Description
<input type="checkbox"/> Goal	RequestPayment	Request pay
<input type="checkbox"/> Action	PaymentRequest	Send payme
<input type="checkbox"/> Percept	PaymentDetailPercept	contains the
<input type="checkbox"/> Goal	ValidatePayment	Check if the
<input type="checkbox"/> Action	PaymentReceipt	Send payme

Figure 2: Example: Get Payment Sub-Scenario Steps

checking the consistency of scenarios when sub-scenarios are used.

All the percepts consumed and actions produced by an agent system need to be attached to one or more scenarios. We also require that a scenario lists all the actions that are generated by the system as part of the execution that is covered by the scenario. Figure 1 is an example of a possible sequence of steps for the book order scenario for an on-line book store, and the steps of its sub-scenario step is shown in Figure 2.

We make three different extensions to the scenario definition in Prometheus. Firstly we define a scenario *IO-sequence* list to capture allowable sequences of percepts and actions for a scenario and its variations. Secondly, we add a parameter based test descriptor that identifies relevant variables for the scenario; and thirdly we add traceability links that propagate scenario information into the detailed design. Each of these extensions are explained below.

Scenario IO-sequence list

We wish to be able to use the scenarios as a basis for an initial form of system testing, that focuses on the percepts coming into the system, and the actions coming out. To assist with this we add an additional structured field to a scenario, which we call the scenario *IO-sequence* list. This list captures all valid sequences of percepts and actions for the particular scenario. We note that each default version of the scenario, as well as each variation, may have multiple allowable IO-sequences as there may be parallelism or non-determinism allowed in some of the orderings. Each valid ordering is specified as a separate IO-sequence in the list. Percepts and actions arising from sub-scenarios must also be represented in each IO-sequence. Figure 3 illustrates the IO-sequence list for the BookOrder scenario.

An initial IO-sequence list, with a single sequence, can be generated by extracting out the percepts and actions as iden-

Type	Name
<input type="checkbox"/> Percept	BookOrderPercept
<input type="checkbox"/> Action	ConfirmStock
<input type="checkbox"/> Action	PaymentRequest
<input type="checkbox"/> Percept	PaymentDetailPercept
<input type="checkbox"/> Action	PaymentReceipt

Figure 3: Book Order Scenario default IO-Sequence

tified in the scenario and its specified sub-scenarios such as in Figure 3. However the designer must add to this list with alternative valid sequences. In addition the IO-sequence list must be specified for scenario variations as we now describe.

The original version of Prometheus scenario descriptors has only one *Variation* field as it is a free text field that can explain multiple variations. In our extension we allow multiple Variation fields, in order to capture separately each logical variation, along with its structured IO-sequence list. Figure 4 shows possible variations for the BookOrder scenario. Variation 1 is for when a book is out of stock. It produces two actions, a notification to the user that the book is out of stock and an order for more stock to the supplier. The two actions may be produced in any order hence there are two possible IO-sequences as shown in the figure.

Figure 4: Example Scenario Variations

When reusing a scenario as a sub-scenario, the developer must consider the alternate scenarios of the sub-scenario in defining the parent level alternate scenarios. Consistency checking can be done by PDT to ensure that the sub-scenario IO-sequences used in the parent scenario, are legitimate sub-scenario sequences. However it is not necessarily the case that all IO-sequences of the sub-scenario will be reflected in the parent scenario or its variations.⁴ Nevertheless it is useful to have the tool alert the user to cases where all the sub-scenario variations are not covered in the parent scenario.

Scenario test descriptor

We also add a *test descriptor* to the scenario design descriptor in order to specify information relevant for testing. This is following the principle of test descriptors used for unit testing in the work of Zhang et al. [12, 11]. The test descriptor is defined based on a set of scenario *variables*, which are parameters identified as important for the particular sce-

⁴Some variations in the sub-scenario may be valid only in contexts other than the parent scenario under consideration.

nario and *initialization procedures* which must be executed prior to triggering the scenario execution.

The scenario variables are described in terms of their type and domain-range, as in Zhang et al. [12]. We also allow a similar specification of relevant relationships between variables as in that work (for example, the number of books ordered is less than the stock available). The test descriptor must also provide the mapping to the implementation.

Figure 5: Book Order Scenario Test Descriptor

In the book ordering scenario, an example variable that would be important is the variable that captures the number of books in stock, *stock_quantity*. Figure 5 shows how this variable is specified as part of the test descriptor, with type *int* and domain range ≥ 0 . It also specifies that the variable belongs at the *agent*-level (opposed to system-level for example) to the *SalesManager* agent class and mapped to the variable *orderBookStock* which is of variable type *simple*. The type of the variable determines how the variable is assigned values at run time. We refer the reader to [12] for further details on this matter.

The initialization procedures are also specified as in Zhang et al. [12]. For example, in Figure 5, a *static* method *initDB*, which is part of the *agent* class *salesManager* is specified to initialize connections to the stock databases which is used during the execution of the scenario.

It is intended that test descriptors are filled out after implementation, although some aspects of important variables may be identified during scenario specification. It is clear at the stage of scenario development that *stock_quantity* is important for the above scenario. However details of the valid range may well be specified after implementation - we may for example choose to have negative numbers indicating number of pending orders for which there is no available stock, or we may choose to have valid values lie between zero and some maximum. The details must be filled in prior to testing, in order to ensure that test cases are generated with all equivalence classes of values for this variable.

Traceability links

Our third modification to scenarios is to introduce links between scenarios and other entities. In the current Prometheus Design Tool (PDT), the design model contains a list of relationship links between entities for traceability. We introduce the following relationship links to ensure traceability of scenarios:

scenario-goal: A link is created between the scenario and the goal that represents the scenario. A link each is also created between the scenario and the goals that are steps of the scenario. These links are used, for example, when a goal is attempted to be deleted, if the goal is part of a scenario

the user is notified of this and prompted to reconsider the deletion or to also delete the goal-step from the scenario. Similarly, when a goal-step is deleted from a scenario the user is asked if the goal representing that goal-step is also to be deleted from the model.

scenario-percept, scenario-action: A link each is created between the scenario and its action and percept steps. This link is used when a percept or action is deleted to prompt the user to also delete the respective scenario step, or reconsider the deletion. When an action or percept is deleted from a scenario, if it is not part of any other scenario then the user is asked if it is to be deleted from the model as well.

goal-event: When a goal is mapped to an event following a process described in the following section, a relationship link is created between the goal and the corresponding event and the goal is annotated with the event that represents it. We note that this is a new field of the goal entity type that we introduce in PDT as it necessary for traceability as follows. If the event of the goal-event link is deleted then the user is notified that the event is associated with the goal and requested to specify an alternative event that represents the goal. If the user provides such an event then the link is modified and the goal is annotated accordingly.

When a goal is deleted, if there is a link to an event (this will be true for all goals assigned to an agent via roles) then the user is prompted to check if the event is also to be deleted⁵.

The goal-event link also enables the consistency check of ensuring the scenario and its goal-steps are mapped to an event.

3. FROM SCENARIOS TO IMPLEMENTATION

In this section we describe how scenarios are mapped from design to implementation in the Prometheus Design Tool (PDT) for the purpose of traceability. We note that we only describe the process and rules for propagation relevant to scenarios and any new techniques that we introduce. We refer the reader to [8] for details on the other aspects. In order to better illustrate the process we use the example of an electronic bookstore similar to that used in [8] with a single scenario of a user ordering a book as illustrated in the previous section, for simplicity.

System Specification In the current methodology, Scenarios are created in the System Specification stage. They are first identified in the Analysis Overview Diagram, where the actors⁶, actions and percepts are specified. Each percept, action and actor is associated with a scenario that responds to the percept, produces the action and interacts with the actor (see Figure 6).

These scenarios are then detailed in the Scenarios descriptor where the trigger and steps of each scenario are identified (Figure 1 details the BookOrder Scenario). Further, as described in the previous section, the IO-sequence lists for the default case and any variations are specified. The scenarios

⁵We note that here we only outline the reasoning around scenarios as these are the additions we propose. There are however, many other rules such as when a goal is deleted, the user is prompted to check if all its sub-goals are also to be deleted and so on.

⁶Entities external to the system that interact with it.

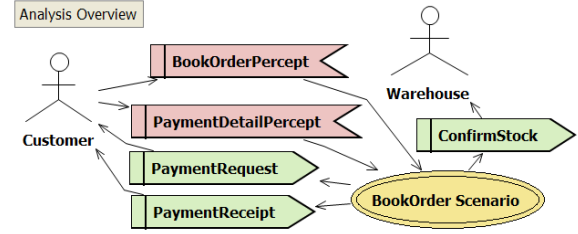


Figure 6: Analysis Overview

and their steps (including the actions and percepts that are part of the IO-sequence lists of variations) are propagated to the level of implementation as follows.

When a scenario is created a corresponding Goal (scenario goal) is automatically created and added to the Goal diagram. This is to ensure that the system has a goal to fulfill the scenario. The Goal Diagram captures the goals of the system. These goals may be further decomposed into subgoals which maybe an “AND” or “OR” decomposition, where “AND” requires all subgoals be satisfied for the goal to be successful and “OR” requires at least one of them to be satisfied. (See Figure 7.)

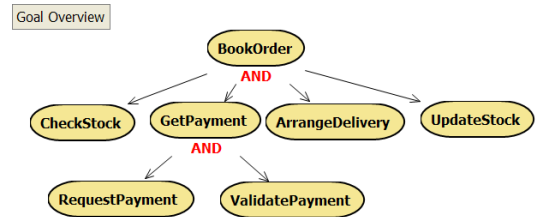


Figure 7: Goals Diagram

For each *goal-step* of a scenario a corresponding goal is automatically created and associated as a subgoal to the scenario goal in the Goal Diagram. Multiple goal-steps are added as an “AND” decomposition as steps are not optional in a scenario.

Action and percepts are typically created in the Analysis Overview Diagram and used by scenarios as steps. However, new actions and percepts may be created when describing the steps of a scenario, which would cause the actions and/or percepts to be automatically added to the Analysis Overview Diagram as associated with the Scenario.

Each *scenario step* follows the same process as above, with the exception that its corresponding goal is associated as a subgoal of the top level scenario goal.

The goals, actions and percepts are propagated to the Roles Diagram where roles are created and associated with them. We note that the steps of a single scenario may be associated with different roles and that not all goals may be assigned to a role, as the goals may be abstract goals. However, all goals must be *covered* by a role, where a goal is considered to be covered if:

- it is explicitly mentioned in the goal descriptor in the design; or
- all of its children are covered; or
- its parent is covered.

This recursive definition ensures a “coverage cut” through the goal hierarchy, where all the goals in the cut are associated with some role and some scenario. Figure 8 is an example of role assignment for the bookstore example.

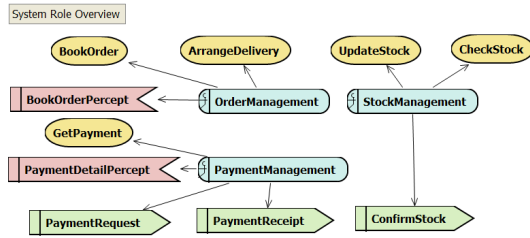


Figure 8: Roles Diagram

We also note that, when a goal is assigned to a role, the subgoals of that goal are by default assigned to the same role, unless they are explicitly assigned to another role.

Architectural Design In the Architectural Design stage, the roles are associated with agents in the Agent-Roles Diagram (see Figure 9). Consequently, the scenario goals and the steps of the scenarios are associated with agents. Note that if a goal is an abstract goal, it is associated with its sub-goals and therefore the agents that the sub-goals belong to.

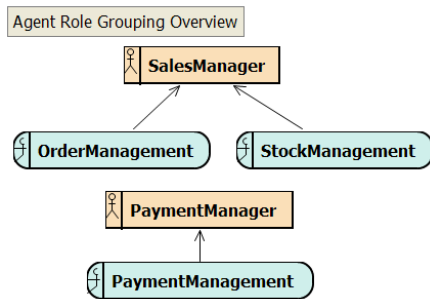


Figure 9: Agent-Role Diagram

The System Overview Diagram illustrates the agents and the percepts that the agent responds to and the actions it produces (see Figure 10).

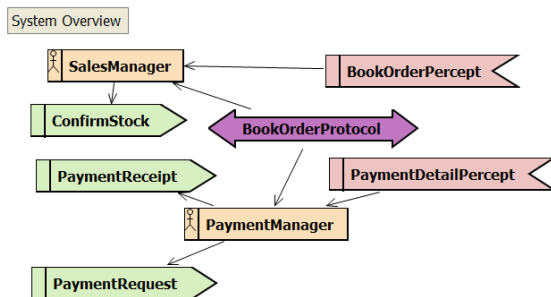


Figure 10: System Overview Diagram

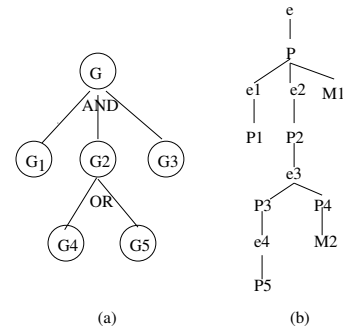


Figure 11: Goals to Events/Plans

Detailed Design

The actions and percepts associated with each agent are propagated into the Agent Overview diagram where the designer then develops plans to handle the percepts and produce the actions.

Currently in PDT goals are not propagated into the Detailed Design, however, the designer may annotate the plans to indicate which goal it satisfies. We extend PDT to automatically propagate goals into the Agent Overview Diagram as described below. The aim is to guide the designer in developing plans to satisfy its goals using the goal hierarchies in the Goal Diagram in such a way that it follows the BDI principles of allowing a choice of plans to achieve a particular goal. This also allows goals to be traceable throughout the design and consequently scenarios, which are mapped to goals, to also be traceable.

For each (top-level) goal associated with an agent (which is automatically determined from the Agent-Role Diagram, and stored in the agent descriptor), depending on the goal decomposition attained from the Goal Diagram, corresponding (empty skeleton) plan and event structures are created in the Agent Overview Diagram as follows:

An event is created to represent the goal, which we term the goal-event, and;

- If the goal has no decomposition then a plan is created to handle the goal-event. The designer may add further plans which represent the different ways of achieving the goal.

- If the goal has “AND” decomposed subgoals (e.g. G has G1, G2 and G3 in Figure 11(a)), a plan is created to handle the goal-event (e.g. P in Figure 11) which posts the subgoals as follows:

If the subgoal is associated with the same agent, a corresponding (subgoal) event is posted by the plan, and a plan that handles that event is created (e.g. event e1, and plan P1 in Figure 11(b)). The designer may add further plans to handle the event, if there is more than one way of achieving the subgoal.

If the subgoal is associated with another agent, then a message is created as outgoing from the plan (e.g. M1 in Figure 11(b)) and the message is propagated into the Agent-Overview Diagram of the other agent.

- If the goal has “OR” decomposed subgoals (e.g. G2 has G4 and G5 in Figure 11(a)) then a plan is created for each

subgoal so that the designer may encode the choice between the subgoals as context conditions of the plans (e.g. P3 and P4 in Figure 11(b)). This supports the BDI principles in programming agents.

For each subgoal that is part of the same agent, an event is posted by the subgoal plan, and a plan that handles the event is created (e.g. e4 and P5 in Figure 11(b)). The reason for posting an event to handle the goal rather than handling the goal as part of the plan that posts it, is to allow the designer to add alternate ways of achieving the subgoal if any.

For each subgoal that is part of another agent, a message is created as outgoing from the plan (e.g. M2 in Figure 11(b)) and propagated into the AgentOverview Diagram of the other agent.

A similar process is applied recursively down the tree until all the goals are either mapped to an event and at least one plan that handles that event, or a message that is delegated to another agent. Figure 12 shows the AgentOverview Diagram for the SalesManager agent which incorporates the propagation process described.

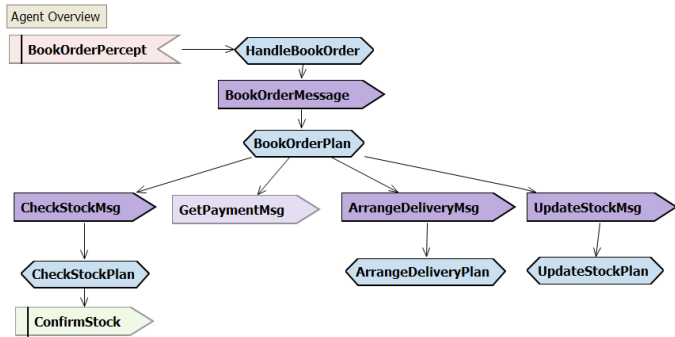


Figure 12: Agent Overview Diagram: SalesManager

Consistency Cross-Check for Scenarios:

PDT provides a *consistency check* feature that checks the model against a set of rules and alerts the user if any rule is violated. The following rules ensures that Scenarios and all their steps are mapped to the detailed design which is used to generate the code for the system (PDT provides an automated code generation feature that generates skeleton code in the JACK agent language). For a given scenario:

- All action and percept steps are associated with at least one plan. This includes the actions specified as alternative outputs of the scenario.
- The scenario goal and all its subgoals (as specified in the Goal Diagram) are associated with a role. A goal is associated with a role if one of the following is satisfied:
 - the goal is assigned to a role.
 - all its subgoals are associated with a role: note this is recursive and that different subgoals may be associated with different roles.
 - the goal's parent goal is assigned to a role: by default a goal is assigned to the same role as its parent unless explicitly assigned to another.

- All goals that are part of the scenario and assigned to a role are mapped to an event and the event is handled by at least one plan (this is automated via the propagation process described above, however, the check is to ensure that changes made by the designer does not violate it)
- If a percept is a trigger to the scenario then it must be associated with the same role associated with the scenario goal or, if the scenario goal is abstract, the role associated with the first non-abstract goal-step.
- Each (sub)scenario step also satisfies the above.

4. TEST FRAMEWORK

One of the motivations for the greater structure in the scenario descriptors is to be able to do testing of scenarios as part of requirements or acceptance testing. At this level we want to initiate a particular interaction (scenario) in a wide range of different situations with respect to input variations, and ensure that the system behavior is as expected. “As expected” in this context is defined precisely by the sequences of actions and percepts defined in the IO-sequence lists for the scenario, including its variations. Testing then requires that the environment exists, and perhaps is initialized (e.g. stock levels of books established), the trigger is provided to start off the scenario, and the ensuing sequence of actions and percepts are recorded.

The following analysis may be performed from the recorded sequence of percepts and actions:

- If the sequence matches one of that in the IO-sequence lists for the scenario and its variations, then the test succeeds.
- If a sequence is observed which has not been specified, then a trace of the program, using a tool within the implementation platform, can help in identifying where the sequence has diverged from what was expected. It may also be the case that the unexpected sequence is valid and wasn't identified by the developer in the design, in which case the scenario specification be revised to include the IO-sequence as a variation to the scenario.
- The IO-sequences observed by the different successful test cases are noted. If there are some specified IO-sequences that are not observed, then the tester is notified of this. This could be due to (a) insufficient test cases to cover all possibilities, (b) a fault in the implementation (for example, no plan produces an action that is part of the sequence), or (c) a fault in the IO-sequence specification (for example, the sequence is not achievable).

It is often the case that it is not practical to test the system “in situ”. In this case we need some kind of mechanism to collect the actions and provide the percepts. To achieve this we propose the use of an agent based simulation platform such as Repast [6]. These simulation platforms already provide a sound and well-adopted infrastructure for simulating complex environments such as those that agent systems typically operate in.

We leverage these concepts to provide the basic infrastructure necessary for testing scenarios which we now describe.

The simulator acts as an environment for generating percepts and consuming actions. In other work we have integrated a BDI agent platform (JACK) with an agent based simulation platform (Repast) via percepts and actions with message passing that synchronises the two systems. In the integrated model, each BDI agent that interacts with the simulation environment has a corresponding simulation agent that acts as a *sensor-actuator* for sending percepts from the environment to the BDI agent and executing actions of the BDI agent within the simulated environment (See Figure 13). While it is possible to achieve the same with a single simulation agent, having individual simulation agents provide an independent thread of execution for each agent and also simplifies the implementation.

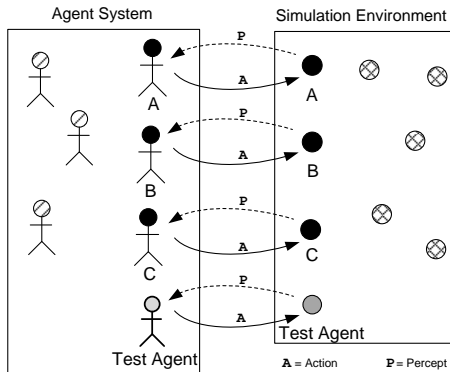


Figure 13: Interaction with the Sim environment

For the purpose of testing the system, we introduce a BDI Test Agent (BDI-TA) and a Simulator Test Agent (SIM-TA), to set up and execute the test cases. Whilst the BDI-TA will be able to directly access and set-up any system level information, in order to set-up information that is internal to an agent, each agent is required to have a *HandleTestData* plan which is triggered by a *TestData* message sent by the BDI-TA.

An overview of the testing process is as follows:

- The SIM-TA sends the relevant test data, such as the initial values for the test variables and initialization procedures to the BDI-TA via a *StartTest* percept which is handled by the *HandleStartTest* plan in the BDI-TA. This marks the start of a test case execution. The test descriptor for the scenario to be tested is used to determine the appropriate information that needs to be initialized.
- The BDI-TA sets up any system level information and runs initialization procedures and sends a *TestData* message to the relevant agents for setting up agent internal beliefs or variables. For example, in testing the BookOrder scenario, the value for the variable *stock_quantity* and the *initDB* initialization procedure will be established by the *SalesManager* agent (refer Figure 5). When all initializations are complete, the BDI-TA informs the SIM-TA of this by sending the action *StartSimulation*.
- The SIM-TA also provides the simulator agents that may generate percepts during the scenario execution

with information to be contained in the percept, which corresponds to a particular test case. It is possible to identify which agent handles the triggering percept for the particular scenario from the design specification using the traceability links as described in section 2.1. For example, in the BookOrder scenario the data required by the *SalesManager* agent to populate the *BookOrderPercept* such as *order_details* will be provided by the SIM-TA. These state variable initialization is achieved by calling the appropriate “set methods” of the simulation agents.

- Once the above set-up process is complete (including receiving the action *StartSimulation* from BDI-TA) the SIM-TA starts the execution of the rest of the simulation agents. For example in the BookOrder scenario, this will make the *SalesManager* simulation agent send the *BookOrderPercept* to the *SalesManager* BDI agent. Once the system execution starts the SIM-TA does not interact with any of the agents in the system as part of the usual execution, but simply records actions generated and percepts received.
- Any action produced by an agent is executed via its corresponding simulator agent, which simulates the effects of the action on the environment. Which may in turn produce percepts. For example, the *PaymentManager* agent produces the *PaymentRequest* action, which when executed in the simulator generates the *PaymentDetailsPercept* which is delivered back to it.
- Apart from action triggered percepts, the scenario may also require percepts at particular time intervals, or when a particular state of the environment is true. Simulator agents execute routines at every simulation cycle. Therefore, the tester/developer should encode the routines for producing these percepts within the corresponding simulator agents. It might also be the case that not sending an intermediate percept is part of a test case that tests the behavior of the system when the percept is not received (does the system fail gracefully?).
- The test case execution ends when a pre-defined timeout is reached for the complete scenario execution. It could also end if a pre-defined timeout for a particular action is not met.
- All the actions and percepts generated during the execution of a particular test case is recorded. The data for all test case executions for a particular scenario is gathered and later analyzed.

Note that we currently only support the testing of a single scenario at a time. We assume the execution of the scenario under test in isolation where interacting scenarios are not considered. Testing interacting scenarios is part of the future work.

In order to assist the developer in the above process the following automated code generation can be performed:

- Using the design specification it is possible to obtain the percepts and actions relevant to a particular agent. The code stubs for executing these actions and generating the percepts are created in the corresponding simulator agent which the developer can then fill-in.

- When generating test cases, the initialization procedures and variables are extracted from the test descriptor of the scenario and presented to the user as a test specification. The user can then set the values of the variables that form the different test cases.
- The code stubs of the *SIM-TA*, *BDI-TA* and the *HandleTestData* plan for each agent are created.

The specification of the variables that influence a scenario in the test descriptor of that scenario allows automated test case generation, using the domain range for the variables and any comparative relationships specified. Such an approach is already fully implemented in PDT for the unit testing framework of Zhang et al. [12].

5. DISCUSSION/CONCLUSION

In this paper we have extended the specification of scenarios in Prometheus methodology to include information that allows (a) a structured specification of the variations of a given scenario in terms of percept and action sequences; (b) traceability of the scenario throughout the various design stages; and (c) the scenario to be tested for the expected outcomes in terms of the specified percept and action sequences. We then provide a detailed process for mapping scenarios to implementation, using the Prometheus Design Tool, automatically propagating information where possible. Finally, we provide an approach for testing the scenarios, using an agent based simulation platform.

Given the nature of an agent system behavior, which includes asynchronous handling of percepts from the environment and simultaneous execution of multiple intentions (plans), it is not trivial to use standard test automation frameworks for testing agent systems. Moreover, existing scripting based automation tools [3] require extensive set up to simulate a complex environment that an agent system operates in. Therefore, we use an agent simulation platform such as Repast for simulating the environment for the agent system and enabling the test case execution for scenarios.

While formal verification is one approach that can be used for validating requirements (such as the Z specification), these methods require additional knowledge in the developers part to apply them and are also not well supported by practical agent implementation platforms (such as JACK) based on main stream programming languages (such as Java). Hence, even if the design model is verified, the implementation should still be tested via a practical approach as proposed in this work.

The testing approach developed compliments other work on testing agent systems such as the unit testing framework of Zhang et. al. [12] which we build upon, the goal-oriented testing approach of the eCAT tool [5] which is based on the goal models associated with the Tropos methodology [4], the JAT framework [2] for testing agents developed in the JADE platform⁷ that specifies a fault model based on general agent features and provides skeleton code for testers to manually develop test cases and the SUNIT framework based on the SEAGENT model [10].

One of the limitations of the current approach is that it tests a given scenario in isolation. This simplification has helped us focus on building a framework for using scenarios for system testing. However, in a “live” agent system

simultaneous scenarios are in execution giving rise to action-percept sequences influenced by each other. More work is required in supporting the test developer and extending the simulation environment to be able to support interacting scenarios.

To our knowledge this is the first work that explores the use of an agent simulation platform as a test automation tool to test an agent system. We intend to explore this further in two directions. Firstly, we intend on evaluating our framework by testing several agent systems from different application domains in order to gain further insight into the effectiveness of our approach. Secondly, we plan to extend the PDT design interfaces to allow developers to graphically define the simulation agent details, providing a unified interface for designing system(BDI) and test(SIM) agents.

6. REFERENCES

- [1] Steve S. Benfield, Jim Hendrickson, and Daniel Galanti. Making a strong business case for multiagent technology. In *Proceedings of AAMAS'06*, pages 10–15, New York, NY, USA, 2006. ACM.
- [2] Roberta Coelho, Uir Kulesza, Arndt von Staa, and Carlos Lucena. Unit Testing in Multi-Agent Systems using Mock Agents and Aspects. In *Proceedings of the 2006 International Workshop on Software Engineering for Large-Scale Multi-Agent Systems*, pages 83–90. ACM Press, 2006.
- [3] Mark Fewster and Dorothy Graham. *Software test automation: effective use of test execution tools*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1999.
- [4] Fausto Giunchiglia, John Mylopoulos, and Anna Perini. The tropos software development methodology: processes, models and diagrams. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 35–36, New York, NY, USA, 2002. ACM.
- [5] Cu D. Nguyen, Anna Perini, and Paolo Tonella. ecat: a tool for automating test cases generation and execution in testing multi-agent systems (demo paper). In *7th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Estoril, Portugal, May 2008.
- [6] Michael J. North, Nicholson T. Collier, and Jerry R. Vos. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Trans. Model. Comput. Simul.*, 16(1):1–25, 2006.
- [7] Lin Padgham, John Thangarajah, and Michael Winikoff. Prometheus design tool. In *Proceedings of The AAAI Conference on Artificial Intelligence*, pages 1882–1883, Chicago, USA, 2008.
- [8] Lin Padgham and Michael Winikoff. *Developing Intelligent Agent Systems: A practical guide*. Wiley Series in Agent Technology. John Wiley and Sons, 2004.
- [9] Leon Sterling and Kuldar Taveter. *The Art of Agent-Oriented Modeling*. The MIT Press, 2009.
- [10] Ali Murat Tiryaki, Sibel Öztuna, Oguz Dikenelli, and Riza Cenk Erdur. Sunit: A unit testing framework for test driven development of multi-agent systems. In *AOSE*, pages 156–173, 2006.
- [11] Zhiyong Zhang, John Thangarajah, and Lin Padgham. Automated unit testing intelligent agents in pdt. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1673–1674, Estoril, Portugal, 2008.
- [12] Zhiyong Zhang, John Thangarajah, and Lin Padgham. Model based testing for agent systems. In *10th International Workshop on Agent Oriented Software Engineering (AOSE2009)*, Budapest, Hungary, May 2009.

⁷jade.tilab.com/

Kokomo: An Empirically Evaluated Methodology for Affective Applications

Derek J. Sollenberger
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
djsollen@ncsu.edu

Munindar P. Singh
Department of Computer Science
North Carolina State University
Raleigh, NC 27695-8206, USA
singh@ncsu.edu

ABSTRACT

The introduction of affect or emotion modeling into software opens up new possibilities for improving user experience. Yet, current techniques for building affective applications are limited, with the treatment of affect in essence handcrafted in each application. The multiagent middleware Koko attempts to reduce the burden of incorporating affect modeling into applications. However, Koko can be effective only if the models it needs to function are suitably constructed.

We propose Kokomo, a methodology that employs expressive communicative acts as an organizing principle for affective applications. Kokomo specifies the steps needed to create an affective application in Koko. A key motivation is that Kokomo would facilitate the construction of an affective application by engineers who may lack a prior background in affective modeling.

We empirically evaluate Kokomo's utility through a developer study. The results are positive and demonstrate that the developers who used Kokomo were able to develop an affective application in less time, with fewer lines of code, and with a reduced perception of difficulty than developers who worked without Kokomo.

Categories and Subject Descriptors

D.2.8 [Software Engineering]: Design—*Methodologies*; I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*Multiagent systems*

General Terms

Design, Experimentation, Human Factors

Keywords

Software engineering, Affective computing, Computational architectures for learning

1. INTRODUCTION

The term *affect* is used in psychology to refer to feelings or emotions. The term is also used more narrowly to describe an *expressed* or *observed* emotional response of a human to some relevant event. Expressed responses can be most

Cite as: Kokomo: An Empirically Evaluated Methodology for Affective Applications, Derek J. Sollenberger and Munindar P. Singh, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 293-300.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

easily demonstrated in a 3D virtual environment where virtual characters use gestures and dialog to emulate human emotional responses [7]. In contrast, we concentrate on the *observation* aspects of affective computing. In particular, we focus on enabling software developers to use affective computing techniques in order to create a superior user experience.

We take as our point of departure the Koko middleware developed by Sollenberger and Singh [17, 19]. Koko is a service-oriented middleware that observes and maintains a predictive model of a user's affective state, and thus relieves application developers from the challenge of maintaining such a model. However, Koko needs to be suitably configured so as to work effectively.

We propose a methodology called *Kokomo* that steps an application developer through a process for configuring Koko and incorporating it into an application. Interestingly, Kokomo is based on the notion of *expressives*, an important but little-known (especially in the agents community) class of communicative acts.

Further, we conducted a developer study to determine the actual and perceived benefits of Koko and Kokomo to application developers. Our hypothesis is simple: developers who use Koko and Kokomo can more easily construct an affective application than those who do not, while at the same time not diminishing the quality of their application. The study consisted of the same application assigned to groups of developers employing Kokomo (with Koko), Koko alone, and neither (just traditional techniques).

Our evaluation measured the ease of constructing an application both subjectively and objectively. We collected subjective developer feedback on their perceived difficulty via surveys which we obtained throughout the duration of the study. The feedback indicates a lower perception of difficulty for the affective portions of the assignment when using Kokomo than without it. Analysis of objective difficulty measures (code metrics and effort analysis) shows that Kokomo yields the best results on nearly every measure of code complexity and effort. However, the results were not uniformly strong for developers employing Koko alone (we revisit this point in Section 6).

It is important to note, that this work does not introduce any new theories or paradigms for the field of affective computing, but rather seeks to employ existing affective theories in a compelling new way. The main contributions of this work are focused on software engineering as we seek to expand the scope of agent-oriented methodologies into the realms of expressive communication and affective agents.

Additionally, unlike many agent-oriented methodologies, we have subjected our methodology to an empirical evaluation in the form of a developer study where the developers are not the authors of the methodology.

2. BACKGROUND

Smith and Lazarus [16] are credited with developing a theory of emotions, called *appraisal theory*, which has become the baseline model for affect in computational systems. As Figure 1 shows, an appraisal (for our purposes, an expression of affect) arises from how a user interprets or *construes* an event in the environment.

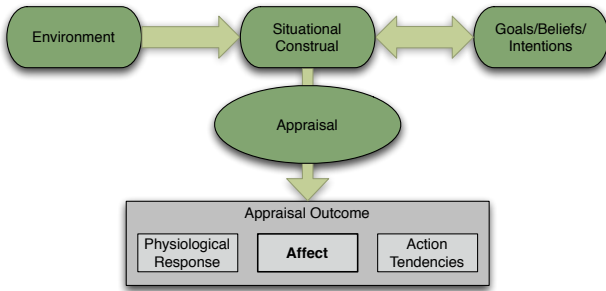


Figure 1: Components and process of an appraisal.

Further, the affective state contained within an appraisal outcome can be interpreted as a vector of discrete states, each with an associated intensity. For instance, upon successfully completing a task, a user could arrive at an appraisal that he or she is simultaneously *happy* with an intensity of α and *proud* with an intensity of β .

Appraisal theory finds broad application because of its computational nature. Its applications include educational games [1], similar to the task in our developer study. We note that affective computing extends beyond our discussion of appraisal theory. Picard [12] provides an excellent overview of the affective computing field which includes a more detailed discussion of appraisal theory.

A major challenge in applying appraisal theory is its domain dependence: it yields models of affect that are tied to a particular domain and context. The common practice when creating a new affective application have been to copy and modify an existing model to meet the specifications of the new domain. Recent approaches have begun to address this challenge. The EMotion and Adaptation (EMA) system [7] for modeling virtual characters incorporates a domain-independent model to be populated with the necessary context at runtime. CARE [10] does the same for modeling human subjects.

2.1 Koko

Koko is a multiagent and service-oriented middleware, which enables the prediction of a user’s affective state [17]. Koko is not an affect model, but a platform within which (existing and future) models of affect can operate.

Koko provides an ontology of emotions and an ontology of the primitives for describing an application’s state. A developer uses these ontologies to configure Koko by specifying (1) which emotions to model and (2) the relevant components of the application’s state. From the configuration, Koko generates a model specific to the application. At runtime Koko receives inputs, described using the configured

primitives, from the sensors and the application.

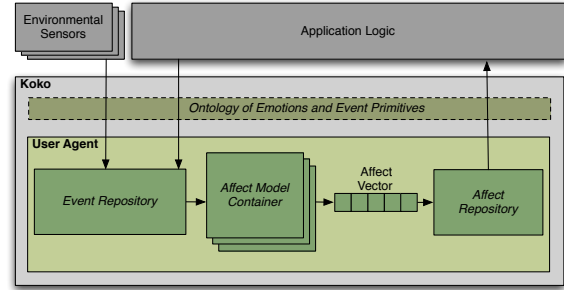


Figure 2: A Koko user agent (arrows represent data flow).

Koko supports appraisal theory models, as described earlier in this section. Appraisal theory dictates that each time an input is received an output must be produced. The output produced by a Koko model is a vector of probabilities, called an affect vector. The vector contains an element for each emotion as configured for the application. Each element is the probability that the user is currently experiencing the corresponding emotion. Koko makes each user’s affect vector available to the appropriate applications.

Koko follows the cognitive-based user affect modeling (CB-AUM) approach [8]. Koko supports one model per application for each user and exposes these models to applications using a formal set of interfaces [17]. A CB-AUM model relies heavily on machine learning techniques. Koko includes a repository for CB-AUM models from which a developer may choose the most appropriate model for a new application. The basic flow of such a model can be conceptualized in the following steps.

1. Seed the model with information about the user’s environment. This seeding is based on the application-specific configuration.
2. Provide training data. Either the user or someone observing the user must record the user’s emotional state in conjunction with data from the application or the sensors, preferably both. This is simplest if the application itself is appropriately instrumented, e.g., to query the user for their emotions. The data can optionally be acquired offline.
3. Learn a model from the training data. Koko does this using the Weka toolkit. In general, user affect modeling [8] makes heavy use of various machine learning techniques to adapt to the behavior patterns of individual users.
4. Provide probabilistic predictions regarding the user’s affective state, represented in Koko as an affect vector, as described above.

Additionally, Koko is designed to support affective interactions among the user agents. The traditional Koko runtime is deployed in a cloud environment where there is one Koko agent per end user regardless of how many different Koko based applications are associated with the user. These agents are equipped to share affective information with other agents in the user’s social circle, even across application boundaries.

2.2 Koko-ASM

Koko-ASM is a methodology for configuring a social (multiagent) affective application using Koko [18]. The configuration process of Koko is important because the affect models contained in Koko are dependent on the quality of information they receive from the application. The purpose of the methodology is to guide the application developer through Koko’s configuration process in an intuitive manner.

Instead of simply listing the requirements for Koko’s configuration, Koko-ASM uses the concept of *agent* interaction to guide the developer through the process. The methodology makes use of speech act theory [15] as a means of modeling communication between agents in a social environment. The primary focus of the methodology is to identify the communicative actions among agents and then through a series of steps decompose those interactions into the artifacts needed to configure Koko.

3. KOKOMO

We now describe Kokomo, our proposed configuration methodology for Koko. Kokomo addresses the same goal as Koko-ASM, namely, to guide the application developer to configure Koko for an application. Koko-ASM and Kokomo share some commonalities: both are based on agents, both use the concept of agent roles, and both employ speech act theory.

However, Kokomo has some critical and fundamental differences in terms of how it achieves its goal. First, although Koko is designed to operate in environments ranging from those that are highly social to those containing only a single agent, Koko-ASM only considers applications that involve a high degree of interaction among Koko’s user agents. Kokomo, in contrast, is broader in its scope and applies to all settings where Koko can be used. Second, Kokomo expands its treatment of agents beyond a Koko user agent (standing in for a user) to include both Koko user agents and virtual AI-controlled entities (e.g., virtual character). The third and final difference is that no two steps are equivalent across the methodologies. In fact, Kokomo adds two additional steps to provide more granular guidance for developers.

Upon completion of a design exercise with Kokomo, a developer will have identified or constructed all the artifacts needed to configure Koko for an application. Table 1 lists the steps used to create an affective application with Koko. The documentation below highlights the key concepts of Steps 1–5, which are the primary steps in Kokomo.

Step 1 requires the developer to identify the set of roles an agent may assume in the context of the application. Examples of such roles include teacher, student, coworker, enemy, and rival. A single agent can assume multiple roles and a role can be restricted to apply to the agent only if certain criteria are met. For example, the role of coworker may only apply if the two agents communicating work for the same organization.

Step 2 requires the developer to describe the expressive messages or *expressives* exchanged between various roles [15]. Searle defines expressives as communicative acts that enable a speaker to express their attitudes and emotions toward a proposition. Examples include statements like “Congratulations on winning the prize!” where the attitude and emotion is congratulatory and the proposition is winning the

Table 1: The main steps of Kokomo.

#	Description	Artifacts Produced
1	Define the set of possible roles an agent may assume	Agent Roles
2	Describe the expressives exchanged between roles	Expressive Messages
3	Derive the modeled emotions from the expressives	Emotions
4	Describe a set of nonexpressive application events	Application Events
5	Construct the appropriate event definitions	Event Definitions
6	Select the sensors to be included in the model	Sensor Identifiers
7	Select the affect model from Koko’s repository	Model Identifier
8	Register with Koko’s runtime environment	Source Code

prize. Formally, the structure of an expressive is

$$\langle \text{sender}, \text{receiver}, \text{type}, \text{proposition} \rangle$$

The *type* of the expressive refers to the attitude and emotion of the expressive and the *proposition* to its content, including the relevant events. The *sender* and *receiver* are selected from the set of roles defined in Step 1. The developer then formulates the expressives that can be exchanged among agents assuming those roles. In the case that both the selected roles can only be assumed by artificially intelligent agents then formulate expressives only for communicative acts that are observable by a user agent. The result is a set of all valid expressive messages allowed by the application.

Step 3 requires the developer to select a set of emotions to be modeled from Koko’s emotion ontology. The selected emotions are based on the expressives identified in the previous step. To compute the set of emotions, we evaluate each expressive involving a user agent and select the most relevant emotions from the ontology for that particular expressive. The selected emotions are then added to the set of emotions required by the application. This process is repeated for every expressive and the resulting emotion set is the output of this step.

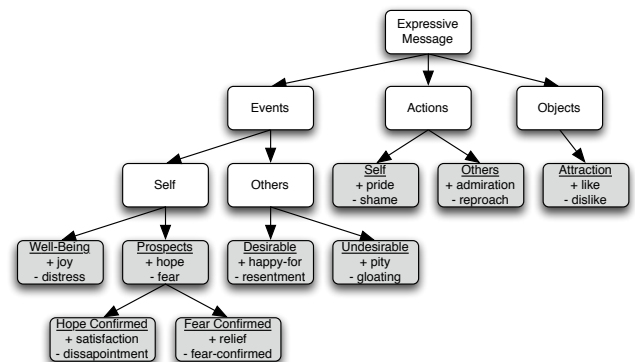


Figure 3: Expressive message hierarchy.

Koko offers support for expressives because it provides a well-delineated representation for affect. Koko can thus exploit a natural match between expressives and affect to help designers operationalize the expressives they use in their ap-

plications. The recommended approach to selecting an emotion is to structure Elliott’s set of emotions [4] as a tree (Figure 3). Each leaf of the tree represents two emotions, one that carries a positive connotation and the other a negative connotation. Given an expressive, you start at the top of the tree and using its *type* and *proposition* you filter down through the appropriate branches until you are left with only the applicable emotions. For example, say that you have a message with a *type* of *excited* and a *proposition* equal to “I won the game.” Now using the tree you determine that winning the game is an action the user would have taken and that *excited* has a positive connotation, so the applicable emotion must therefore be pride. In general, the sender and receiver would have different interpretations of the same event. For example, if the recipient of the above message is the agent who lost the game, then the emotions that are relevant to the recipient would be admiration or reproach depending on their perception of the winner.

If the *proposition* of the expressive message is composite or even ambiguous as to whether or not the *type* applies to an event, action, or object, then more than one path of the tree may apply. Such is the case when an agent conveys its mood via an expressive message. Mood is an aggregation of emotions and therefore does not have a unique causal attribution. For example, an expressive might convey that a agent is generally *happy* or *sad* without being *happy* or *sad* at something. Therefore, we do not select any specific emotion when evaluating an expressive pertaining to mood as the emotions that comprise the mood are captured when evaluating the other expressives. In other words, mood is not treated directly upon the reception of an expressive.

Step 4 requires the developer to describe a set of nonexpressive events in their application. A nonexpressive application event occurs when the user has an interaction with the application that may effect their emotions. In particular, we are interested in the emotions selected in Step 3. The nonexpressive aspect of these events is that they are not a direct result of interaction between two users, but rather of the user with their environment.

A developer could encode the entire state of the application as a set of events, but this is not necessary as we are only interested in the parts of the application that may effect the emotions we have selected. For example, the time the user has spent on a current task would likely influence their emotional status, whereas the time until the application needs to clear its caches or garbage collect its data structures is likely irrelevant.

We guide developers by helping them think about the interactions (direct and indirect) a user can have with their application and how those interactions relate to the set of emotions defined in Step 3. For example, if the application monitored phone calls and the user had not sent or received a call all day then that may affect the user’s emotional state. Further, after identifying an event we must quantify it. In the previous example, the quantification could be the time since the last call was received.

Step 5 requires the developer to construct the appropriate event definitions using Koko’s event ontology. The events described are a combination of the expressives in Step 2 and nonexpressive application events identified in Step 4. Each expressive identified in Step 2 is modeled as two event definitions, one for sending and another for receipt. The decomposition of a message into two events is essential because we

cannot make the assumption that the receiving agent will read the message immediately following its receipt and we must accommodate for its autonomy.

Step 6 and **Step 7** both have trivial explanations. Koko maintains a listing of both the available sensors and affect models, which are accessible by their unique identifiers. The developer must simply select the appropriate sensor and affect model identifiers.

Step 8 completes the methodology by providing the developer with details on how to register their application within Koko’s runtime environment. Given the artifacts gathered from the previous steps we can configure the application via the interfaces defined by the Koko middleware. Upon success, the Koko registration interface returns an *applicationID*, which acts as the identifier for the application. The developer uses the *applicationID* in all subsequent interactions with the Koko runtime, such as sending information about the user to Koko or querying for the user’s current affective state.

4. APPLICATION FOR OUR STUDY

Our study involves the development of a math tutoring application for high-school students. This application centers around a dynamic lesson planning agent called the *virtual teaching assistant (vTA)*. The goal of the vTA is to sharpen the user’s mathematical skills in a variety of areas, such as probability and geometry. The target audience is students preparing for the standardized end-of-grade tests that are given to US students at the conclusion of the 8th and 12th grades.

This application satisfies three essential criteria. First, it is simple enough to implement in a short period of time, yet complex and open enough that multiple solutions are possible. Second, the application can be compartmentalized into discrete units, thereby enabling us to compare individual components across all developers. Third, the education domain lends itself to affective interactions, where the role of affect in learning is well documented [14].

Let us further explain the role of affect in the learning process. The primary motivation comes from Csikszentmihályi’s theory of flow [2]. In highly simplified terms, the ideas is that individuals learn best in situations where they are neither bored (because the challenge appears trivial) nor frustrated (because the challenge appears impossible), but instead in an intermediary state called the *flow channel*. This concept has successfully (from an education standpoint) been applied in virtual learning environments [11].

4.1 Application Details

Figure 4 describes our application. At all times the user interface is under the control of either the virtual teaching assistant (vTA) or the testing algorithm. The remaining boxes represent the various services that are available to be used by either the vTA or the testing algorithm.

The expected application usage is as follows. A user can elect to start a tutoring session from a welcome screen. A tutoring session is comprised of 15 mathematical problems where the first ten problems are selected by the vTA and the remaining five by a testing algorithm. The goal of the vTA is to select the ten best training problems in order to prepare the user for the test. The vTA has at its disposal a database of all possible questions as well as a set of physical sensors that can provide information on the user’s current state.

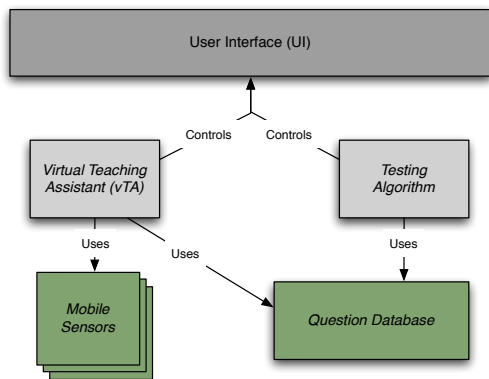


Figure 4: Application components.

Using those sensors and by collecting other data such as the user's history, environment, and current affective state the vTA crafts a customized dynamic lesson plan for each user. The application presents the user with a series of problems (defined by the plan), who is then tested via the testing algorithm as a means of evaluating the effectiveness of the vTA's plan.

Each developer is provided working code for all components of the application with the exception of the vTA. In this manner, we reduce nonaffect-related variability in the solutions and in our measurements of quality and effort, and force our developers to focus on the logic of the application. Using the theory of flow as motivation, the developers are responsible for designing and implementing their own customized vTA.

Now we describe each component of the application from the developer's perspective. Some components are used by the vTA and others serve as an outlet for information produced by the vTA.

Problem Set (Question Database)

The problem set is a subset of a question bank that is maintained by the National Center for Educational Statistics (<http://nces.ed.gov>). We extracted all multiple choice mathematics questions asked to 8th through 12th grade students based on data collected by the National Assessment of Educational Progress.

Each problem in the dataset contains an *ID*, *question*, *grade level (targeted)*, *content area* (properties and operations, geometry, analysis and probability, or algebra), *difficulty* (percentage of students who correctly answered the question on national standardized tests), and *answer*. The questions are text based, some with a greyscale illustration, and include five multiple choice options as answers.

A developer can retrieve a set of questions based on any permutation of the problem components in a manner analogous to SQL queries, with which our intended developers are familiar. Or, a developer can request a randomly selected problem that meets specified criteria, e.g., a random geometry problem that fewer than 20% of students answered correctly.

User Interface

As Figure 4 shows, the vTA controls the user interface (UI). The developer determines the UI via predefined control points, which help limit developer effort and provide uniformity across implementations. Figure 5 shows the two primary

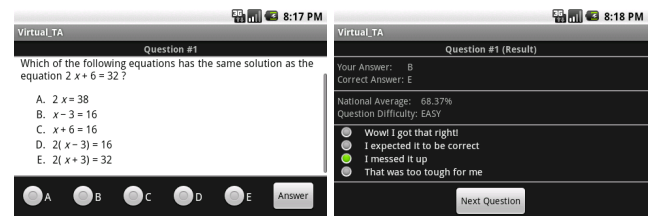
screens of the four screens in the UI. Using these basic screens the vTA may configure the user experience with the following four actions.

Pre-Session Questionnaire. Just before a training session begins, the vTA may present the user with a set of up to three questions, and use the answers received to initialize the lesson plan for the user.

Ask Question. After initialization, the UI requests a question from the vTA, who responds with the ID of a question, which the UI displays on the screen.

Record Result. The UI conveys the user's answer to the vTA and shows the user the correct answer.

Post-Question Feedback. After each question, the vTA can optionally ask the user one multiple choice affect-related question, which can provide the vTA with better insight into the user's current affective state.



(a) Question Presentation (b) Result and Feedback

Figure 5: The application's user interface.

The vTA controls the user interface for the first ten problems to train the user, at which point the testing algorithm assumes control and asks the final five questions. The testing algorithm interacts the same as the vTA, but may additionally display the most recent user's history on the application's start page.

Mobile Sensors

A unique advantage of building a mobile application is the availability of sensory inputs. Most smart-phones today come equipped with sensors such as GPS, accelerometer, microphone, and proximity sensors. Access to data provided by these sensors is useful for determining the environment in which the user is operating as well as for evaluating the emotional state of the user. Using this data, developers can track the performance of users in specific environments and customize the lesson plan accordingly.

We employ Android as our development platform. Each developer had access to an Android phone with real sensor inputs. Android provides utility classes for a variety of sensors to use interfaces for extracting data from the sensors.

Virtual TA

The virtual TA is at its core an intelligent agent that maintains a reasoning engine. It is in this engine where the developer must use the available tools, such as sensors and the question database, to craft a unique lesson plan for the user. The vTA has one simple goal: *Create the lesson plan for the user that gives them the best chance of success when tested.*

In general, a vTA could carry out complex reasoning, such as based on the user's previous standardized test scores and

available national statistics. However, allowing such extensive approaches would have introduced too many variables into the evaluation. To reduce the variability, we instructed all developers to consider only three categories of information (exam, environmental, and emotional data) in their reasoning algorithms to formulate a lesson plan.

Exam data is any data that can be computed from the user's past performance. It is up to the developer how complex the analysis of the exam data is, but they must at least consider the previous question and the overall performance of the user on the previous exam.

Environmental data is any data gathered from a physical sensor. For instance, by using the microphone on the device a developer could monitor how well the user performed in noisy environments. The developers must integrate data from at least one sensor into the reasoning logic of their vTA.

Emotional data is qualitative feedback from the user on their current status, which can be obtained after each math question as a means to determine how the user felt about the last question. A developer would use this feedback along with the theory of flow as a basis for updating the lesson plan. Developers are required to use such feedback to influence the next question as well as store the feedback for future analysis when selecting questions.

The developer must construct a reasoning algorithm that uses the data to generate a lesson plan. In addition to writing the algorithm, each developer was required to provide written documentation detailing how the algorithm functioned.

Testing Algorithm

The testing algorithm selects the five questions used to evaluate the user at the end of each training session. The algorithm considers the question category, difficulty, and recurrence, and asks no more than two questions from any category. Therefore, since there are only four categories, the test always covers a minimum of three categories. The algorithm also ensures that the national average for correctly answering at least one question is above 70% correct (easy) and one is below 30% (hard). Finally, the algorithm ensures that no question is repeated in the next three exams for the user.

The details of this algorithm were provided to the developers so they were aware of the testing strategy. Further, we store each user's past performance on exams and make it available to the vTA.

5. EVALUATION

We selected 30 students to participate in the developer study. Each developer had experience programming in Java, and had no prior experience in affective computing. The developers were paired into teams of two and given basic programming assignments in order to evaluate their programming abilities. The assignments were evaluated by a third-party reviewer who ranked each team based on its programming proficiency.

Additionally, each developer was given a survey asking them to rate their software engineering experience and describe projects they had worked on. Over 97% of the developers had a minimum of two years experience programming in Java, but for 88% of developers the initial programming assignments were their first exposure to programming in the

Android operating system.

Using the results of the developer surveys and the programming assignments, we ranked each developer with respect to their peers. Using those rankings, we then divided the teams into three groups so as to equalize the estimated programming ability across the groups. Each team within a group was assigned a variation of the same task. The control group did not have access to Koko or Kokomo, one of experimental groups had access to Koko, and the second experimental groups had access to Koko and Kokomo.

All groups are given identical instructions regarding vTA. The Koko and Kokomo groups were additionally provided instructions for using Koko and Kokomo, respectively. Each team was given four weeks in which to complete the assignment. They had to design and develop the vTA agent using either the components provided (e.g., question database and user interface) or any API provided by the Android SDK.

We conducted a baseline quality check for each application. This check ensured that all applications ran appropriately and did not omit any of the features in the required feature set. This test was done in order to strengthen our claim that the introduction of Koko and Kokomo does not diminish the quality of the application. All applications from all three groups passed the baseline check. In fact, we observed that applications using Koko and Kokomo had additional features, in particular, affective features that were not present in the control groups applications. However, our statistical measures disregard such extra features.

Each team was evaluated in the same manner regardless of its group. We evaluated the study in both objective and subjective ways. The objective portion involved measuring the time spent by each developer and various measurements on the source code. The subjective portion was based on periodic surveys of the developers to measure their perceptions regarding their feelings on the complexity of each aspect of the project and the utility of the tools provided.

In-study surveys. These were completed by each developer every time they worked on the project. A minimum of one survey was required for each day (and within one day) the developer worked on the project. The survey collected information such as the time spent on each component of the application as well as their perceptions and comments on difficulty.

Post-study surveys. There were completed by each developer at the end of the development cycle to describe their perception of the entire project. Additionally, this survey was used as a mechanism for developers to recommend improvements to the project and the tools.

The remainder of this section focuses on the evaluation of our hypothesis. Our hypothesis stated that developers who use Kokomo can more easily construct affective application than those who do not, while at the same time not diminishing the quality of their application. The following two sections evaluate the objective and subjective metrics respectively. Finally, after viewing each category of metrics independently we take a holistic view of the results to determine if our claim is satisfied.

5.1 Objective Results

Measuring the complexity of a project based on the resulting source code is nontrivial because there are no definitive

measures of complexity. Complexity can relate to the size of the code base and also to less measurable notions such as extensibility and maintainability. Here we adopt some well-known metrics from software engineering [5, 13], and use them to analyze each project independently. Finally, we compare the metrics across all three developer groups. Figure 6 highlights the results for four of the most well-known metrics. In all cases except Figure 6(d), the lower value indicates a better result.

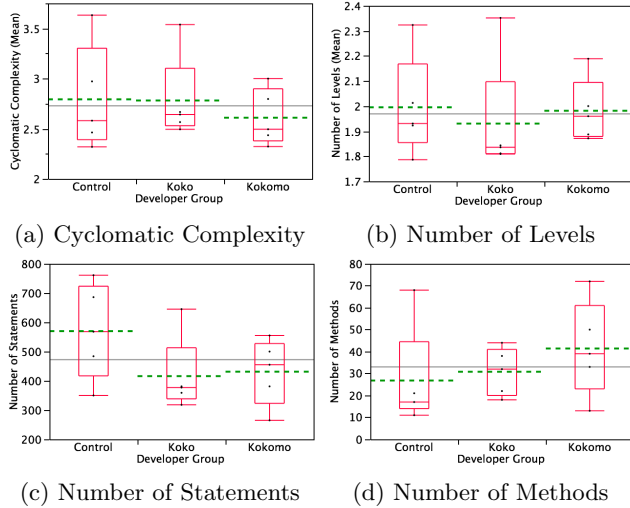


Figure 6: Software metrics from our study.

CC McCabe’s Cyclomatic complexity [9] indicates the number of “linear” segments in a method (i.e., sections of code with no branches) and therefore helps determine the number of tests required to obtain complete coverage. It also indicates the psychological complexity of a method.

NoLm The number of levels per method reflects the number of logical branches each method has on average. NoLm is a key factor in determining code readability, and is also used to determine how well the code adheres to in object oriented design patterns.

NoS The number of statements in the project is a common measure of the amount of time spent developing and the general maintainability of the code.

NoM The number of methods in the project reflects increasing modularity and readability of the code (for a fixed NoS value).

We had the developers log the time they spent on each portion of the assignment. Figure 7 gives an overview of the time spent on the project as a whole as well as its affective components specifically. It is important to note that though the total project time did not drastically decrease with Koko, the percentage of time spent on affect fell sharply. Additionally, the drastic reduction of time and minimal variance for Kokomo in Figure 7(c) illustrate the benefit of the methodology.

5.2 Subjective Results

Several survey questions asked the developer to either rank components against one another or rate the individual

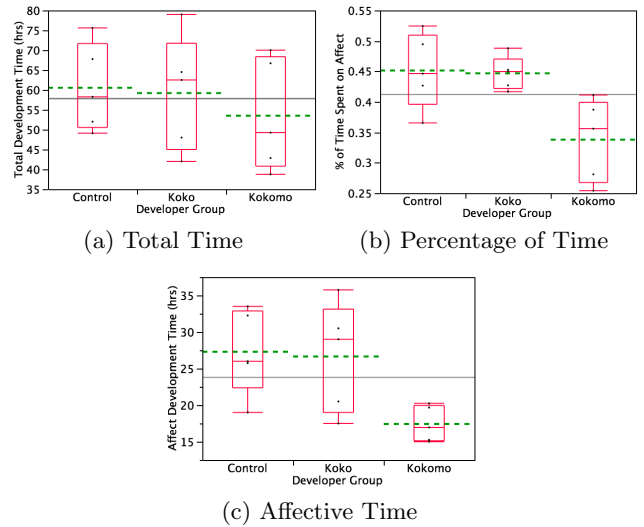


Figure 7: Development time metrics.

components on a difficulty scale. Table 2 shows the average difficulty rankings given by each development group for the five key components of the project. You can see that the introduction of both Koko and Kokomo reduced the developer’s perception of difficulty for the affective aspects of the project.

Table 2: Difficulty ranking of project components.

Rank	Control	Koko	Kokomo
1 (Easiest)	Heuristic	Heuristic	<i>Affect</i>
2	Programming	<i>Affect</i>	Heuristic
3	Exam	Programming	Environment
4	<i>Affect</i>	Environment	Programming
5 (Hardest)	Environment	Exam	Exam

Further, we asked developers to individually rate each component of the assignment on a difficulty scale of 1 (least) to 5 (most). As shown in Figure 8(a), 90% of the developers using Kokomo rated the difficulty of the affective component as a 1 or 2. However, 70% of the control development group gave the same component a rating of 3 or higher. Finally, Figure 8(b) shows how developers rated the benefit obtained from the Koko tools on a benefit scale of 1 (none) to 5 (high). Of the developers using Kokomo, 70% perceived Koko to offer a high benefit to them (a rating of 4 or 5) whereas only 40% of those without Koko gave a similar rating.

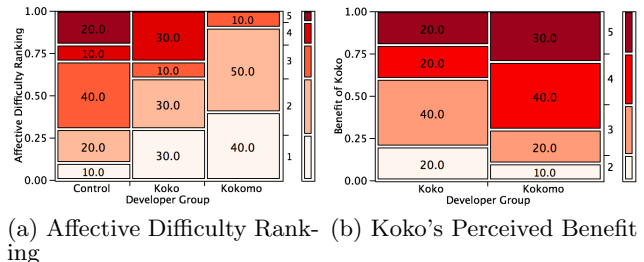


Figure 8: Subjective developer distributions.

5.3 Conclusions

Using both objective and subjective data, we have empirically shown that the introduction of Kokomo results in both

perceived and tangible benefits for developers. With respect to software metrics the applications written using Kokomo had either equivalent or better results than those written without Kokomo. The use of Kokomo resulted in significant time savings and also significantly reduced the time variance for constructing the affective component. The reduction of the time variance is important in that it enables developers to more accurately budget their time. More accurate time estimates in turn enable them to better scope the cost of incorporating affect into an application.

Further, in all cases where developers were asked their perceptions of the project those using Kokomo had a more positive perception than those who did not. This positive perception of Kokomo combined with the tangible benefits make a compelling case for its practical adoption by software developers.

6. DISCUSSION

The most perplexing result of our study was that the data showed that the introduction of Koko without Kokomo did not result in a marked improvement. Upon evaluating the source code and developer surveys, we realized that several developers who were only given Koko were not able to completely grasp the affect modeling concepts that underlie Koko. As a result, they were unsure of how to configure the Koko middleware. Whereas the above observations strengthen the case for Kokomo's usefulness, we plan to improve our documentation regarding Koko for future studies and deployment.

Existing agent-oriented software engineering methodologies specify messages at a high level and therefore are not granular enough to support the expressive communication employed by Kokomo [3]. Further, Kokomo's applicability has a much narrower scope than these methodologies since it is restricted to affective applications that may or may not employ multiagent systems in their implementation. These distinctions are simply the result of a difference in focus. It is quite possible, given the narrow scope of Kokomo, that it could be integrated with broader methodologies in order to leverage their existing processes and tools. For example, many methodologies have detailed processes by which they help developers identify all possible messages that are exchanged among agents. Kokomo would benefit by integrating such processes, thereby making it easier to identify the expressive messages.

Further, model-driven development methodologies, such as INGENIAS [6] or ASEME [20], are relevant. Kokomo could benefit from such methodologies by using a specific agent modeling description language to describe the expressive interactions among agents. Such a model could be used to produce the source code needed to configure the Koko runtime automatically. This would reduce the amount of time spent translating the expressive messages into a format understood by Koko and would also facilitate improved agility in developing such as in changing the agent interaction model without having to rewrite the source code.

Koko is a multiagent middleware supporting affective interaction among agents, but this study involved only a single agent. This choice was intentional in that it enabled us to demonstrate that unlike Koko-ASM, the Kokomo methodology could be used in single agent environments. In the future, we plan to demonstrate the utility of Kokomo in multiagent settings by using it in social applications that

take full advantage of Koko's social design.

7. REFERENCES

- [1] C. Conati and H. Maclaren. Modeling user affect from causes and effects. In *User Modeling, Adaptation, and Personalization 2009*, volume 5535 of *LNCS*, pages 4–15, Berlin, September 2009. Springer.
- [2] M. Csikszentmihalyi. *Finding flow*. Basic Books, New York, 1997.
- [3] S. A. Deloach, M. F. Wood, and C. H. Sparkman. Multiagent systems engineering. *Int'l J. Soft. Eng. Know. Eng.*, 11(3):231–258, 2001.
- [4] C. Elliott. *The Affective Reasoner: A Process Model of Emotions in a Multi-agent System*. PhD, Northwestern, 1992.
- [5] N. E. Fenton. *Software Metrics: A Rigorous and Practical Approach*. PWS Pub, London, 1997.
- [6] I. Garc a-Magarino, J. Gomez-Sanz, and R. Fuentes-Fernandez. Model Transformations for Improving Multi-agent Systems Development in INGENIAS. In *Proc. AOSE*, pages 25–36, 2009.
- [7] J. Gratch and S. Marsella. A domain-independent framework for modeling emotion. *J. Cognitive Systems Res.*, 5(4):269–306, Dec 2004.
- [8] C. Martinho, I. Machado, and A. Paiva. A cognitive approach to affective user modeling. In *Affective Interactions, LNAI 1814*, pages 64–75, Berlin, 2000.
- [9] T. McCabe. A complexity measure. *IEEE Trans. Soft. Eng.*, 2:308–320, 1976.
- [10] S. McQuiggan, S. Lee, and J. Lester. Predicting user physiological response for interactive environments: An inductive approach. In *Proc. AIIDE*, pages 60–65, 2006.
- [11] S. W. McQuiggan, J. P. Rowe, and J. C. Lester. The effects of empathetic virtual characters on presence in narrative-centered learning environments. *Proc. SIGCHI*, pages 1511–1520, 2008.
- [12] R. W. Picard. *Affective Computing*. MIT Press, Cambridge, MA, 1997.
- [13] R. S. Pressman. *Software Engineering*. McGraw-Hill, 6th Edition, New York, 2005.
- [14] P. A. Schutz and R. Pekrun, editors. *Emotion in Education*. Elsevier, Boston, MA, 2007.
- [15] J. R. Searle. *Speech Acts*. Cambridge University Press, Cambridge, UK, 1970.
- [16] C. Smith and R. Lazarus. Emotion and adaptation. In L. A. Pervin and O. P. John, editors, *Handbook of Personality*, pages 609–637, New York, 1990. Guilford Press.
- [17] D. J. Sollenberger and M. P. Singh. Architecture for affective social games. In *Proc. Agents for Games and Simulations, LNAI 5920*, pages 79–94, Berlin, 2009.
- [18] D. J. Sollenberger and M. P. Singh. Methodology for engineering affective social applications. In *Proc. AOSE*, pages 49–60, 2009.
- [19] D. J. Sollenberger and M. P. Singh. Koko: An Architecture for Affect-Aware Games. In *J. AAMAS*, In press.
- [20] N. Spanoudakis and P. Moraitis. Model-driven agents development with ASEME. In *Proc. AOSE*, 2010.

Programming Mental State Abduction

Michal Sindlar *
Intelligent Systems Group
University of Utrecht
michal@cs.uu.nl

Mehdi Dastani
Intelligent Systems Group
University of Utrecht
mehdi@cs.uu.nl

John-Jules Meyer
Intelligent Systems Group
University of Utrecht
jj@cs.uu.nl

ABSTRACT

Many multi-agent system applications involve software agents that reason about the behavior of other agents with which they interact in cooperation or competition. In order to design and develop those systems, the employed programming languages should provide tools to facilitate the implementation of agents that can perform such reasoning. This paper focuses on BDI-based programming languages and proposes a nonmonotonic reasoning mechanism that can be incorporated into agents, allowing them to reason about observed behavior to infer others' beliefs or goals. In particular, it is suggested that the behavior-generating rules of agents are translated into a nonmonotonic logic programming framework. A formal analysis of the presented approach is provided and it is shown that it has desirable properties.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Intelligent agents—*mental state ascription, nonmonotonic reasoning*

General Terms

Design, Theory

Keywords

Modeling other agents and self, Logic-based approaches and methods, Reasoning (single and multiagent)

1. INTRODUCTION

A fundamental principle of multi-agent systems is that they involve multiple agents, often situated in a (virtual) environment where they interact in cooperation or competition. For implementation of individual software agents in a multi-agent system there exists a range of agent programming languages, such as 2APL, GOAL, Jadex and Jason [5, 9], most of which find their roots in the Belief-Desire-Intention (BDI) paradigm of agency [8, 19]. If such agents

*This research has been supported by the GATE project, funded by the Netherlands Organization for Scientific Research (NWO) and the Netherlands ICT Research and Innovation Authority (ICT Regie).

Cite as: Programming Mental State Abduction, M.P. Sindlar, M.M. Dastani and J.-J.Ch. Meyer, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 301-308.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

have the ability to perceive and interpret the behavior of others, then the principle of mental state ascription comes into focus as a possible explanatory abstraction for reasoning about that behavior. This principle has solid theoretical foundations in terms of the *intentional stance* [10], which is encountered throughout A.I. literature in various guises [6, 7, 13, 21, 4]. However, there exist few principled instantiations specific to BDI-based agent programming of explanatory reasoning about the behavior of other agents. This paper presents such an instantiation, a possible application of which is in computer games where virtual characters are to exhibit believable interaction with other virtual characters. Multiple requirements for character believability (as found by Loyall [17]) are fulfilled by BDI-based software agents (e.g. proactiveness, resource-bounded agency, adaptivity), and some of those which are not (particularly those concerning social interaction) can be met using our approach. This is desirable as it has been shown that characters' believability (and players' enjoyment) increases if characters appear to incorporate mental state ascription into their decision-making [16], and specifically for role-playing games user feedback furthermore indicates that players consider characters to be deficient in this regard [1].

Because the work which is the foundation of this paper is logic-based, and because most agent programming languages utilize logic programming, it is natural that the implementation presented in this paper is a logic program. And since explanatory reasoning is in general defeasible, answer set programming [12] is employed as it is the state-of-the-art programming paradigm for nonmonotonic reasoning. Most systems for answer set programming share their syntax with Prolog, a language which is incorporated into several current agent programming languages (e.g. [9]), making integration of answer set programming natural from a syntactic point of view. In select cases this integration already exists [18].

The work presented in this paper is based on our earlier work on explanatory ascription of mental states to software agents [21], and is presented in the following steps: first, the theoretical underpinnings are briefly explained in Section 2; second, this existing work is formalized in classical logic to lay the foundation for our implementation (Section 3); and third, this implementation is presented and analyzed (Sections 4–6). Sections 7 and 8 conclude with a reflection on related work and our own, respectively.

2. PRELIMINARIES

In this section our work on mental state abduction [21] is restated, in relation to an agent programming language that

contains constructs shared by most state-of-the-art agent programming languages (cf. [5, 9]).

2.1 Agent Programming

This section describes the agent programming language APL used in this paper. The behavior of agents is generated by goal achievement rules of the form $n : \gamma \leftarrow \beta \mid \pi$, stating that the rule with identifier n is suitable for achieving goal γ if β is believed, in which case the plan π can be selected by the agent. The BNF grammar of rules is given below, where it should be noted that it concerns ground programs (i.e. without variables), which suffices for our purposes.

$$\begin{aligned} \langle \text{library} \rangle &::= \langle \text{pgrule} \rangle +; \\ \langle \text{pgrule} \rangle &::= \langle n \rangle \text{“:”} \langle \text{gq} \rangle \text{“<-”} \langle q \rangle \text{“|”} \langle \text{plan} \rangle; \\ \langle \text{gq} \rangle &::= \langle \text{atom} \rangle \mid \langle \text{gq} \rangle \text{“and”} \langle \text{gq} \rangle \mid \langle \text{gq} \rangle \text{“or”} \langle \text{gq} \rangle; \\ \langle q \rangle &::= \langle \text{literal} \rangle \mid \langle q \rangle \text{“and”} \langle q \rangle \mid \langle q \rangle \text{“or”} \langle q \rangle; \\ \langle \text{literal} \rangle &::= \langle \text{atom} \rangle \mid \text{“not”} \langle \text{atom} \rangle \\ \langle \text{plan} \rangle &::= \langle \text{action} \rangle \mid \langle \text{test} \rangle \mid \langle \text{seq} \rangle \mid \langle \text{cond} \rangle \mid \langle \text{iter} \rangle; \\ \langle \text{test} \rangle &::= \text{“B(”} \langle q \rangle \text{“)”} \mid \text{“G(”} \langle q \rangle \text{“)”} \mid \langle \text{test} \rangle \text{“and”} \langle \text{test} \rangle; \\ \langle \text{seq} \rangle &::= \langle \text{plan} \rangle \text{“;”} \langle \text{plan} \rangle; \\ \langle \text{cond} \rangle &::= \text{“if”} \langle \text{test} \rangle \text{“then”} \langle \text{plan} \rangle \text{“else”} \langle \text{plan} \rangle; \\ \langle \text{iter} \rangle &::= \text{“while”} \langle \text{test} \rangle \text{“do”} \langle \text{plan} \rangle \end{aligned}$$

The semantics of rule interpretation are omitted for space conservation and also because the above is a theoretical agent programming language. Nevertheless, this language is subsumed by some existing agent programming languages and is strongly related to others, such that a substantiated discussion of rule interpretation can be given.

In most agent programming languages interpretation of rules occurs as part of a *deliberation cycle* (or ‘sense-reason-act’ cycle), in which the applicability of rules is considered in relation to the configuration of the agent (i.e. its beliefs, goals, intentions, etc.). An agent has a library of rules at its disposition, and if a rule is applicable then it can be fired such that the accompanying plan is adopted by the agent and becomes the agent’s active plan. A plan describes behavior, composing primitive and test actions by means of standard programming constructs (sequence, choice, iteration). It is assumed with regard to rule interpretation that 1) the agent executes the actions of its active plan until this plan is finished or dropped (in which case it is not active anymore); 2) the agent executes actions sequentially (i.e. not concurrently); and 3) the agent has at most a single active plan (i.e. no interleaving of actions from different plans).

2.2 Mental State Abduction

Mental state abduction [21] computes a set of explanations for a software agent’s observed behavior, based on knowledge of its rules. It employs the abductive scheme $\{\phi \rightarrow \psi, \psi\} \approx \phi$, which is further clarified in Section 3.1, in inferring beliefs and goals in relation to rules, plans and observed behavior. In order to do so a relation is established between observed actions and plans, and for this purpose a propositional language and a process language [3] are defined here that allow us to formally describe the rules of the agent program. Those languages are assumed to have a common ground with the agent program; specifically they have shared sets of atomic actions Act and propositions Atom , respectively. The propositional language \mathcal{L}_0 and process lan-

guage \mathcal{L}_Π are then defined through $\phi \in \mathcal{L}_0$ and the typical element $\pi \in \mathcal{L}_\Pi$, as follows, given $p \in \text{Atom}$ and $\alpha \in \text{Act}$.

$$\begin{aligned} \phi &::= p \mid \neg\phi \mid \phi_1 \vee \phi_2 \\ \pi &::= \alpha \mid \mathbf{B}\phi? \mid \mathbf{G}\phi? \mid \pi_1; \pi_2 \mid \pi_1 + \pi_2 \mid \pi^* \end{aligned}$$

The operator $?$ defines (unobservable) test actions which can be composed by sequential composition, non-deterministic choice, and iteration ($;$, $+$, and $*$, respectively), along with observable actions $\alpha \in \text{Act}$.

It is here assumed that for APL programming rules of the form $n : \gamma \leftarrow \beta \mid \pi$ holds that $n \geq 1$, $\gamma, \beta \in \mathcal{L}_0$ and $\pi \in \mathcal{L}_\Pi$. Just like in [21], we treat such rules as implications for the purpose of abduction. The preconditions γ, β of a rule $n : \gamma \leftarrow \beta \mid \pi$ are considered abducible, and can be abduced if observed actions are related to π . To relate primitive actions to a plan, *observable sequences* which are generated by this plan are considered. Given $\alpha \in \text{Act}$ the language of observables \mathcal{L}_Δ is defined through its typical element $\delta ::= \alpha \mid \delta_1 \delta_2$ to consist of sequences of actions, such that the function $OS : \mathcal{L}_\Pi \rightarrow \wp(\mathcal{L}_\Delta)$ translates complex expressions (which may involve tests and looping or branching constructs) to sets of observable sequences, as follows.

$$\begin{aligned} OS(\alpha) &= \{\alpha\} \\ OS(\mathbf{B}\phi?) &= \emptyset \\ OS(\mathbf{G}\phi?) &= \emptyset \\ OS(\pi_1; \pi_2) &= OS(\pi_1) \bullet OS(\pi_2) \\ OS(\pi_1 + \pi_2) &= OS(\pi_1) \cup OS(\pi_2) \\ OS(\pi^*) &= \bigcup_{n \in \mathbb{N}} OS(\pi^n), \text{ where } \pi^0 = \text{skip} \ \& \ \pi^{n+1} = \pi; \pi^n \end{aligned}$$

The composition operator $\bullet : \wp(\mathcal{L}_\Delta) \times \wp(\mathcal{L}_\Delta) \rightarrow \wp(\mathcal{L}_\Delta)$ takes arguments $\Delta_1, \Delta_2 \subseteq \mathcal{L}_\Delta$ and maps them to $\{\delta_1 \delta_2 \mid \delta_1 \in \Delta_1, \delta_2 \in \Delta_2\}$ if $\Delta_1 \neq \emptyset \ \& \ \Delta_2 \neq \emptyset$, to Δ_1 if $\Delta_2 = \emptyset$, to Δ_2 if $\Delta_1 = \emptyset$, or to \emptyset otherwise. Note that $OS(\text{skip}) = \emptyset$.

In this paper we focus on the case of *complete observation*, in which all actions of an agent are observed. The relation characterizing this condition is the prefix relation on observable sequences $\preceq = \{(\delta, \delta), (\delta, \delta\delta') \mid \delta, \delta' \in \mathcal{L}_\Delta\}$, because if all actions are observed then an observed sequence must be the prefix of the observable sequence of some plan. Mental state abduction under the assumption of complete observation is then functionally implemented by $\text{msa}_{\mathcal{R}}$, defined as follows, based on a set \mathcal{R} of APL rules.

$$\text{msa}_{\mathcal{R}}(\delta) = \{(\gamma, \beta) \mid \exists(n : \gamma \leftarrow \beta \mid \pi) \in \mathcal{R} \exists \delta' \in OS(\pi) : \delta \preceq \delta'\}$$

In the next section mental state abduction is cast in the mold of classical abduction, which is introduced in Section 3.1.

3. MENTAL STATE ABDUCTION VIEWED AS CLASSICAL ABDUCTION

This section presents an account of mental state abduction in terms of classical logical abduction, focusing on the relating between rule application on the one hand, and action sequences and the agent’s mental state on the other.

3.1 Classical Abduction

Abduction in classical logic is typically considered in the context of *explanation* [2]. Given a logical theory Θ and an observed fact O , under certain conditions a hypothesis H can be abduced which explains the observation. The fact

that H is abduced as explanation for O with respect to Θ is denoted $\Theta, O \approx H$, and defined as follows in terms of classical entailment \models , where $\text{cl}(\Phi)$ denotes the consequential closure of the set Φ ; i.e. all facts logically following from Φ .

$$\Theta, O \approx H \text{ iff } \Theta \cup H \models O \ \& \ \Theta \not\models O \ \& \ H \not\models O \ \& \ \Theta \cup H \not\models \perp \\ \& \ \neg \exists H' : (\text{cl}(H') \subset \text{cl}(H) \ \& \ \Theta \cup H' \models O)$$

The definition of \approx varies per context, but the above is often encountered [2]. Note that the last clause expresses the *minimality* of H : there should not exist a hypothesis H' such that the consequential closure of H' is a strict subset of that of H , where H' also accounts for O . Also note that the term ‘theory’ is used without requiring closure under cl .

An example of abductive explanation which is regularly found in literature [2, 15] goes as follows. Let $\Theta = \{r \rightarrow w, s \rightarrow w\}$ be a logical theory stating that the grass is wet (w) both if it rains (r) and if the sprinklers are on (s). The grass being wet as such is not accounted for by the theory (i.e. $\Theta \not\models w$) but it would be accounted for if it were the case that it rains or that the sprinklers are on, i.e. $\Theta, \{w\} \approx r$ and $\Theta, \{w\} \approx s$. Note, though, that $\Theta, \{w\} \not\approx r \wedge s$ under the above definition of \approx , because the hypothesis $r \wedge s$ is not minimal. Also note that \approx is nonmonotonic: if it is learned at some point that it rains, such that if $\Theta' = \Theta \cup \{r\}$, then it is evident that $\Theta, \{w\} \approx s$ and $\Theta \subseteq \Theta'$, but $\Theta', \{w\} \not\approx s$.

3.2 Translating Rules

As stated earlier, we (informally) treat APL rules of the form $n : \gamma \leftarrow \beta \mid \pi$ as implications ‘ $\gamma, \beta \Rightarrow \pi$ ’ for the sake of employing the classical abductive syllogism. However, this is formally not correct because it need not hold that the agent selects plan π if it has goal γ and belief β . The reason for this lies in the fact that rules cannot be fired if the agent has an active plan, even if they are in principle applicable. A more precise implicative treatment of rules of the above form is therefore ‘ $n \Rightarrow \gamma, \beta, \pi$ ’, to be interpreted as stating that the application of the rule implies that the agent has a particular mental state and has selected a particular plan. Thus, ‘mental state abduction’ is better viewed as ‘applied rule abduction’, where the abduced rule implies a particular mental state. A logical description of rules can then be provided that allows for treating mental state abduction as a case of classical abduction, which is done in the present section by formulating a logical theory that describes rule application by the agent in relation to its mental state and behavior. It is noteworthy in this respect that a single rule is accompanied by a single plan, but that a single plan can give rise to multiple observable sequences. Specifically, if a plan gives rise to multiple observable sequences then it also gives rise to multiple *computation sequences* [14] (although not necessarily vice versa). Let $CS : \mathcal{L}_\Pi \rightarrow \wp(\mathcal{L}_\Pi)$ be the function that computes computation sequences of plans, and be defined like the function OS except for

$$CS(\mathbf{B}\phi?) = \{\mathbf{B}\phi?\} \quad CS(\mathbf{G}\phi?) = \{\mathbf{G}\phi?\}$$

For technical simplicity, and without loss of generality, it is assumed that individual actions in computation sequences are separated by sequential composition (i.e. the symbol ‘;’) and are therefore in the language \mathcal{L}_Π . The translation from APL program to logic uses the following predicates:

- $\mathbf{r}(n, c)$: The agent has applied rule with number n and performs the computation sequence identified by c .

- $\mathbf{b}(\psi)$: The agent’s belief base entails ψ .
- $\mathbf{g}(\psi)$: The agent’s goal base entails ψ .

It is assumed that the background theory Θ is a subset of ground predicate logic, and that the predicates $\mathbf{b}/1$ and $\mathbf{g}/1$ take as argument the result of a function τ defined as follows, where atoms of the set \mathbf{Atom} are available as constants.

$$\begin{aligned} \tau(p) &= p, \text{ if and only if } p \in \mathbf{Atom} \\ \tau(\neg\phi) &= \mathbf{neg}(\tau(\phi)) \\ \tau(\phi \vee \phi') &= \mathbf{disj}(\tau(\phi), \tau(\phi')) \\ \tau(\phi \wedge \phi') &= \mathbf{conj}(\tau(\phi), \tau(\phi')) \end{aligned}$$

The function τ thus maps (ground) APL terms from a logical form to a functional representation, which is left as-is for now. At a later point (in Section 5.1) this functional representation is utilized in order to regain the truth function of logical connectives, but up to then it can simply be regarded as a string that points to a particular belief/goal precondition (i.e. $\tau(\gamma)$ points to γ). It is more convenient, though, to already define and employ the translation now, in order to save space and effort in later sections.

The translation of a set of rules \mathcal{R} of the form $n : \gamma \leftarrow \beta \mid \pi$ to a logical theory $\Theta_{\mathcal{R}}$ is then given below in Formula 1, where ι is a function that assigns a unique numerical identifier to its argument.

$$\begin{aligned} \forall(n : \gamma \leftarrow \beta \mid \pi) \in \mathcal{R} \ \forall \pi' \in CS(\pi) : \\ \mathbf{r}(n, \iota(\pi')) \rightarrow (\mathbf{g}(\tau(\gamma)) \wedge \mathbf{b}(\tau(\beta))) \in \Theta_{\mathcal{R}} \end{aligned} \quad (1)$$

In order to abduce the applied rule on grounds of observed actions, action observability must be formalized as part of the theory. This is realized using the following predicate:

- $\mathbf{o}(\alpha, n)$: Action α is observable as the n ’th action.

The theory $\Theta_{\mathcal{R}}$ relates rule application to action observability, and it should be noted in this respect that the ‘position’ at which an action is observable is reified in instances of the predicate $\mathbf{o}/2$ which are coupled to rule application by means of the relation expressed below.

$$\begin{aligned} \forall(n : \gamma \leftarrow \beta \mid \pi) \in \mathcal{R} \ \forall \pi' \in CS(\pi) : \\ OS(\pi') = \{\alpha_1 \cdots \alpha_m\} \implies \\ \mathbf{r}(n, \iota(\pi')) \rightarrow (\mathbf{o}(\alpha_1, 1) \wedge \dots \wedge \mathbf{o}(\alpha_m, m)) \in \Theta_{\mathcal{R}} \end{aligned} \quad (2)$$

The set of rules \mathcal{R} is here assumed to be fixed, and for any action α_i which is part of some observable sequence, let $i \in \mathbb{N}$ denote that action’s position in the sequence. The translation of the set of rules \mathcal{R} into the theory $\Theta_{\mathcal{R}}$ then allows for abductively inferring instances of $\mathbf{r}/2$ based on observations, which are defined in the next section.

3.3 Observables and Abducibles

The definition of abductive explanation \approx in Section 3.1 does not explicitly specify the domain of the hypothesis H , which is common in treatises concerning logical abduction. However, in computational approaches to abduction the hypotheses are typically restricted to a set of *abducibles* [15]. This approach is adopted here, and $\mathcal{ABD} = \{\mathbf{r}(n, c) \mid c, n \in \mathbb{N} \ \& \ n \geq 1\}$ is introduced as the set of abducibles. Furthermore, the set $\mathcal{OBS} = \{\mathbf{s}(\alpha, n) \mid \alpha \in \mathbf{Act} \ \& \ n \in \mathbb{N} \ \& \ n \geq 1\}$ of *observables* (i.e. possible observations) is introduced, based on the following predicate.

- $\mathbf{s}(\alpha, n)$: The action α is seen as the n 'th action.

Because the theory $\Theta_{\mathcal{R}}$ generated from the set \mathcal{R} of APL rules, as defined in the previous section, specifies only the actions which are *observable* (as opposed to actually observed, or *seen*), a relation is made between observation and observability using the operator ω , defined as follows.

$$\omega(\{\mathbf{s}(\alpha, n), \dots, \mathbf{s}(\alpha', n')\}) = \mathbf{o}(\alpha, n) \wedge \dots \wedge \mathbf{o}(\alpha', n')$$

The rationale behind the ω -operator is that if some actions have been observed then those actions naturally must be observable, and this latter fact can be utilized to abduce, with respect to some $\Theta_{\mathcal{R}}$, the pair of rule and computation sequence (i.e. some instance of $\mathbf{r}/2$) which accounts for the observability of those particular actions.¹

In order to treat mental state abduction (Section 2.2) as a case of classical abduction, some further refinements to the background theory must be made. This is illustrated with the following example, in which the set of APL rules $\mathcal{R} = \{1 : p \leftarrow p' \mid a; b, 2 : q \leftarrow q' \mid c; d\}$ and theory $\Theta_{\mathcal{R}}$ is as follows, based on Formulae 1 and 2.

$$\Theta_{\mathcal{R}} = \{\mathbf{r}(1, 1) \rightarrow (\mathbf{g}(p) \wedge \mathbf{b}(p')), \mathbf{r}(2, 2) \rightarrow (\mathbf{g}(q) \wedge \mathbf{b}(q')), \\ \mathbf{r}(1, 1) \rightarrow (\mathbf{o}(a, 1) \wedge \mathbf{o}(b, 2)), \mathbf{r}(2, 2) \rightarrow (\mathbf{o}(c, 1) \wedge \mathbf{o}(d, 2))\}$$

Observe that for observation $O = \{\mathbf{s}(b, 2) \wedge \mathbf{s}(d, 2)\}$ holds $\Theta_{\mathcal{R}}, \omega(O) \approx \{\mathbf{r}(1, 1), \mathbf{r}(2, 2)\}$, such that application of both rules 1 and 2 and execution of two computation sequences of the corresponding plans is the *minimal* explanation for this observation. This O states different actions to be observed as the second action, whereas our assumption is that observation is sequential such that only a single observation can occur at any moment. For this reason some restrictions are imposed on any *actual observation* $O \subseteq \mathbf{OBS}$, as follows.

$$\forall \alpha \in \mathbf{Act} \quad \forall n \in \mathbb{N} : \\ [(\mathbf{s}(\alpha, n) \in O) \ \& \ (n > 1) \implies \\ (\exists \alpha' \in \mathbf{Act} : \mathbf{s}(\alpha', n-1) \in O)] \ \& \quad (3) \\ [\forall \alpha' \in \mathbf{Act} : (\mathbf{s}(\alpha, n), \mathbf{s}(\alpha', n) \in O) \implies \\ (\alpha = \alpha')]$$

The constraint of Formula 3 ensure that instances of $\mathbf{s}/2$ in O are numbered consecutively, starting with 1, and that for each n at most a single action is observed.

However, consider $O' = \mathbf{s}(a, 1) \wedge \mathbf{s}(d, 2)$ and verify it to respect the constraint of Formula 3, yet observe that again $\Theta_{\mathcal{R}}, \omega(O') \approx \{\mathbf{r}(1, 1), \mathbf{r}(2, 2)\}$. In order to rule out such explanations, it must be explicitly assumed that the agent's observed behavior stems from a single computation sequence of some plan belonging to a single rule, as follows.

¹An interesting insight, provided by the effort to reformulate mental state abduction as a case of classical abduction, is that mental states can be treated as *observables*, on par with actions. In the case where actions are the observable input to abductive explanation, rules are abduced which account for those actions. Given an input of a mental state as observable — on grounds of the agent communicating its goals and/or beliefs to the observer, for example — rules are abduced which could be applied by the agent given that particular mental state. And, taking this train of thought yet a station further, nothing withstands having both actions and beliefs/goals as observable input, such that abduction of rules which account for the observed actions, given that particular (fragment of the) agent's mental state, occurs.

$$\forall (m : \gamma \leftarrow \beta \mid \pi), (n : \gamma' \leftarrow \beta' \mid \pi') \in \mathcal{R} \\ \forall i \in \{\iota(\pi'') \mid \pi'' \in CS(\pi)\}, j \in \{\iota(\pi''') \mid \pi''' \in CS(\pi')\} : \quad (4) \\ (m \neq n) \implies \neg(\mathbf{r}(m, i) \wedge \mathbf{r}(n, j)) \in \Theta_{\mathcal{R}} \ \& \\ (i \neq j) \implies \neg(\mathbf{r}(m, i) \wedge \mathbf{r}(n, j)) \in \Theta_{\mathcal{R}}$$

Given the above, the following holds.

THEOREM 1. *Let \mathcal{R} be a set of APL rules, theory $\Theta_{\mathcal{R}}$ the smallest set closed under Formulae 1, 2, and 4. Then $\forall H \subseteq \mathbf{ABD} : \Theta_{\mathcal{R}}, \omega(O) \approx H \implies H$ is a singleton (set).*

PROOF. *Consider any two distinct $h, h' \in \mathbf{ABD}$, and let $H \subseteq \mathbf{ABD}$ such that $\Theta_{\mathcal{R}}, \omega(O) \approx H$. If $\{h, h'\} \subseteq H$ but $\neg \exists (h \rightarrow \psi) \in \Theta_{\mathcal{R}}$ or $\neg \exists (h' \rightarrow \psi) \in \Theta_{\mathcal{R}}$, then $\Theta_{\mathcal{R}}, \omega(O) \not\approx H$ because H is not minimal. If $\exists (h \rightarrow \psi) \in \Theta_{\mathcal{R}}$ and $\exists (h' \rightarrow \psi) \in \Theta_{\mathcal{R}}$, then $\neg(h \wedge h') \in \Theta_{\mathcal{R}}$ and $\Theta_{\mathcal{R}}, \omega(O) \not\approx H$ because $\Theta_{\mathcal{R}} \cup H \models \perp$. Thus, H cannot contain two distinct elements and must be a singleton $H = \{h\}$ for some $h \in \mathbf{ABD}$. \square*

3.4 Proof of Correspondence

In this section correspondence is shown between the functional approach of mental state abduction by means of $\mathbf{msa}_{\mathcal{R}}$, as restated in the previous section, and the classical abductive approach based on the theory $\Theta_{\mathcal{R}}$, as put forward in the current section.

THEOREM 2. *Let \mathcal{R} be a set of APL rules, and theory $\Theta_{\mathcal{R}}$ the smallest set closed under Formulae 1, 2, and 4. Then, given $l(\delta) = \{\mathbf{s}(\alpha_1, 1), \dots, \mathbf{s}(\alpha_n, n)\}$ iff $\delta = \alpha_1 \dots \alpha_n$,*

$$\forall \delta \in \mathcal{L}_{\Delta} \quad \forall \gamma, \beta \in \mathcal{L}_0 : \quad (\gamma, \beta) \in \mathbf{msa}_{\mathcal{R}}(\delta) \iff \\ \exists H \subseteq \mathbf{ABD} : \Theta_{\mathcal{R}}, \omega(l(\delta)) \approx H \ \& \ \Theta_{\mathcal{R}} \cup H \models \mathbf{g}(\tau(\gamma)) \wedge \mathbf{b}(\tau(\beta))$$

PROOF. (\implies) *Choose any $\delta \in \mathcal{L}_{\Delta}$ and $(\gamma, \beta) \in \mathbf{msa}_{\mathcal{R}}(\delta)$, and note that $\exists (i : \gamma \leftarrow \beta \mid \pi) \in \mathcal{R} \exists \delta' \in OS(\pi) : \delta \preceq \delta'$ must hold. Choose any such $i : \gamma \leftarrow \beta \mid \pi$ and let $\delta' = \alpha_1 \dots \alpha_n$, so that $\exists(\mathbf{r}(i, j) \rightarrow \mathbf{o}(\alpha_1, 1) \wedge \dots \wedge \mathbf{o}(\alpha_n, n)) \in \Theta_{\mathcal{R}}$ on grounds of Formula 2. Let $\delta = \alpha_1 \dots \alpha_m$ and note that $\mathbf{o}(\alpha_1, 1) \wedge \dots \wedge \mathbf{o}(\alpha_n, n) \rightarrow \omega(l(\delta))$, so $\Theta_{\mathcal{R}} \cup \{\mathbf{r}(i, j)\} \models \omega(l(\delta))$ holds. Thus $\Theta_{\mathcal{R}}, \omega(l(\delta)) \approx \{\mathbf{r}(i, j)\}$, observing that the requirements of \approx are met and $\{\mathbf{r}(i, j)\} \subseteq \mathbf{ABD}$. On grounds of Formula 1 holds $\exists(\mathbf{r}(i, j) \rightarrow \mathbf{g}(\tau(\gamma)) \wedge \mathbf{b}(\tau(\beta))) \in \Theta_{\mathcal{R}}$, such that $\Theta_{\mathcal{R}} \cup \{\mathbf{r}(i, j)\} \models \mathbf{g}(\tau(\gamma)) \wedge \mathbf{b}(\tau(\beta))$.*

(\impliedby) *Take some $\delta \in \mathcal{L}_{\Delta}$ and assume $\exists H \subseteq \mathbf{ABD} \exists \gamma, \beta \in \mathcal{L}_0 : \Theta_{\mathcal{R}}, \omega(l(\delta)) \approx H \ \& \ \Theta_{\mathcal{R}} \cup H \models \mathbf{g}(\tau(\gamma)) \wedge \mathbf{b}(\tau(\beta))$. It has been proven in Theorem 1 that H must be a singleton, say $H = \{\mathbf{r}(i, j)\}$ for some $i, j \geq 1$. Then $\exists(\mathbf{r}(i, j) \rightarrow \mathbf{g}(\tau(\gamma)) \wedge \mathbf{b}(\tau(\beta))) \in \Theta_{\mathcal{R}}$ such that $\exists (i : \gamma \leftarrow \beta \mid \pi) \in \mathcal{R}$, because $\Theta_{\mathcal{R}}$ is the smallest set closed under Formula 1, 2 and 4. Because it is given that $\Theta_{\mathcal{R}}, \omega(l(\delta)) \approx \{\mathbf{r}(i, j)\}$ it must be the case that $\exists(\mathbf{r}(i, j) \rightarrow \psi) \in \Theta_{\mathcal{R}}$ such that $\psi \rightarrow \omega(l(\delta))$. Let $\psi = \mathbf{o}(\alpha_1, 1) \wedge \dots \wedge \mathbf{o}(\alpha_n, n)$, and observe that $\exists \pi' \in CS(\pi) : \iota(\pi') = j$ such that $OS(\pi') = \{\alpha_1 \dots \alpha_n\}$ must then be the case. Let $\delta = \alpha_1 \dots \alpha_m$ with $m \leq n$, such that $\omega(l(\delta)) = \mathbf{o}(\alpha_1, 1) \wedge \dots \wedge \mathbf{o}(\alpha_m, m)$ and, because any output of the function l respects the constraint of Formula 3, it holds that $\delta \preceq \alpha_1 \dots \alpha_n$. If that is the case, then $(\gamma, \beta) \in \mathbf{msa}_{\mathcal{R}}(\delta)$. \square*

4. IMPLEMENTATION

This section presents the implementation of a logic program, based on the logical theory presented in Section 3, assuming use of the grounder LPARSE [22] and the solver CLASP [11], winner of several recent competitions.

4.1 Answer Set Programming

ASP (answer set programming, or Answer Set Prolog [12]) is a logic programming language with answer set semantics, whose main programming construct are rules of the form

$$l_i : - l_{i+1}, \dots, l_j, \text{not } l_{j+1}, \dots, \text{not } l_k. \\ m\{l_1, \dots, l_i\}n : - l_{i+1}, \dots, l_j, \text{not } l_{j+1}, \dots, \text{not } l_k.$$

where each l denotes a literal (i.e. an atom or its strong negation), l_i or $m\{l_1, \dots, l_i\}m$ is the *head* and $l_{i+1}, \dots, \text{not } l_k$ the *body* of the rule. Negation in literals $-p$ is referred to as *strong negation*, and negation in *extended literals* $\text{not } l$ as *default negation*. Rules of the above form can be interpreted (informally) as stating that the head should be satisfied if the body is, where the head is either a single literal or a *choice literal* of the form $m\{l_1, \dots, l_i\}n$ stating that a minimum of m and maximum of n literals of $\{l_1, \dots, l_i\}$ should be in candidate answer sets if the body of the rule is satisfied [22]. If the head of a rule is omitted then it is a *constraint rule*, which (informally) states that a candidate answer set which satisfies the body should be discarded. A rule with an empty body is called a *fact* and is always satisfied.

The crux of ASP semantics is the generation of *minimal sets* that satisfy a program [12]; cf. the minimality criterion in the definition of \models . Essentially, those sets are models of the logic program, and if every answer set of a program \mathcal{P} satisfies some ϕ it is written $\mathcal{P} \models \phi$. The above introduction to ASP is, by necessity, incomplete and superficial. However, literature abounds with practical and theoretical expositions on ASP: a concise and recent overview with pointers to further reading is given by Gelfond [12].

4.2 From Theory to Program

The logical theory $\Theta_{\mathcal{R}}$ presented in Section 3 can be translated to a logic program quite straightforwardly. First of all, implications occurring in the theory are translated as choice rules, stating that each of the conjoined literals in the consequent of the implication should be in the answer set if the antecedent is. Translation to the program $\mathcal{P}_{\mathcal{R}}$ is then as follows, where $\Theta_{\mathcal{R}}$ is a theory based on some set of rules \mathcal{R} , i.e. the smallest set closed under Formulae 1, 2, and 4.

$$\phi \rightarrow (\psi_1 \wedge \dots \wedge \psi_n) \in \Theta_{\mathcal{R}} \implies \\ n\{\psi_1, \dots, \psi_n\}n : - \phi. \in \mathcal{P}_{\mathcal{R}} \quad (5)$$

The translation on grounds of Formula 5 is illustrated below, given $\mathcal{R} = \{1 : p <- p' \mid a; b, 2 : q <- q' \mid c; d\}$ and $\Theta_{\mathcal{R}}$ as in Section 3.3, such that program $\mathcal{P}_{\mathcal{R}}$ is as follows.

$$2\{g(p), b(p')\}2 : - r(1, 1). \\ 2\{o(a, 1), o(b, 2)\}2 : - r(1, 1). \\ 2\{g(q), b(q')\}2 : - r(2, 2). \\ 2\{o(c, 1), o(d, 2)\}2 : - r(2, 2).$$

This program as such does not yet implement mental state abduction under complete observation, though, because there is neither mention of abducibles nor of observables.

4.2.1 Abducibles

In contrast to Section 3, which takes a constructive approach in the sense that a hypothesis H is abduced *if it meets* the requirements of the relation \models , the implementation takes a deconstructive approach in the sense that each hypothesis is considered as possible explanation and ruled

out *if it does not meet* those requirements. Consideration of abducibles as possible explanations is reflected in program $\mathcal{P}_{\mathcal{R}}$ by the following relation with the theory $\Theta_{\mathcal{R}}$.

$$\Theta_{\mathcal{R}} = \{\mathbf{r}(m, i) \rightarrow \psi, \dots, \mathbf{r}(n, j) \rightarrow \psi', \\ \neg(\mathbf{r}(m, i) \wedge \mathbf{r}(n, j)), \dots\} \quad (6) \\ \implies 1\{\mathbf{r}(m, i), \dots, \mathbf{r}(n, j)\}1. \in \mathcal{P}_{\mathcal{R}}$$

Informally, this means that single distinct instances of $\mathbf{r}/2$ are considered as explanations. Theorem 1 shows that an abduced explanation H must be a singleton subset of the abducibles \mathcal{ABD} because of the constraint of Formula 4, which is reflected in the numerical bounds on the choice literal $1\{\mathbf{r}(m, i), \dots, \mathbf{r}(n, j)\}1$ stating that only a single instance of $\mathbf{r}/2$ is considered as explanation. The condition under which some candidate answer sets are discarded is implemented in the following section in relation to observables.

4.2.2 Observables

The translation from theory to program discussed in the previous section shows that instances of $\mathbf{o}/2$ are entailed on grounds of instances of $\mathbf{r}/2$. In Section 3.3 it was explained how abduction takes place on grounds of a set $O \subseteq \mathcal{OBS}$ of observations, respecting the constraints of Formula 3. The predicate $\mathbf{s}/2$ was defined to denote observed (seen) actions, and instances of this predicate are assumed to be *facts* in the answer set. This is achieved in relation to observations as follows, where $\mathcal{P}_{\mathcal{R}}$ is the program derived from theory $\Theta_{\mathcal{R}}$ (the smallest set closed under Formulae 1, 2 and 4 in relation to a set of APL rules \mathcal{R}), and $O \subseteq \mathcal{OBS}$ is an observation respecting the constraint of Formula 3.

$$\forall \mathbf{s}(\alpha, n) \in O : \quad \mathbf{s}(\alpha, n). \in \mathcal{P}_{\mathcal{R}} \quad (7)$$

Note that the constraints of Formula 3 could be implemented as constraints in the answer set program as well, but that this is unnecessary if instances of $\mathbf{s}/2$ are derived from O .

As said, the implementation presented in this paper takes a deconstructive approach by ruling out invalid candidate answer sets. Elimination of candidate answer sets takes place on grounds of the single following constraint, which expresses that a candidate answer set with one or more actions that have been seen but are not deemed observable at that particular step, should be discarded.

$$: - \mathbf{s}(A, T), \text{not } \mathbf{o}(A, T). \quad (8)$$

In Section 3.3 the function ω was defined to consider observed actions in terms of their observability for the sake of abduction; in effect, Formula 8 is the (deconstructive) counterpart of that function.

THEOREM 3. *Let \mathcal{R} be a set of APL rules, $\Theta_{\mathcal{R}}$ the smallest set closed under Formulae 1, 2, and 4, $O \subseteq \mathcal{OBS}$ a non-empty observation, and program $\mathcal{P}_{\mathcal{R}}$ as derived from $\Theta_{\mathcal{R}}$ and O with Formulae 5, 6, 7, and 8. Then*

$$\forall H \subseteq \mathcal{ABD} : \quad (\Theta_{\mathcal{R}}, \omega(O) \models H) \iff$$

$$\mathcal{P}_{\mathcal{R}} \text{ has an answer set } S, \text{ such that } S \models \bigwedge H \wedge \bigwedge O$$

PROOF. (\implies) *Assume the antecedent to be the case, and observe that for each possibly abducible H a candidate answer set exists on grounds of Formula 6. Let $H = \{h\}$, and note that if $\Theta_{\mathcal{R}}, \omega(O) \models H$ then $\exists (h \rightarrow \psi) \in \Theta_{\mathcal{R}}$ such that*

$\psi \rightarrow \omega(O)$. Note that, on grounds of Formula 5, ψ is represented in the candidate answer set S for which $\bigwedge\{h\} \in S$, and, since $\psi \rightarrow \omega(O)$, that Formula 8 does not rule out S , which means that S is a valid answer set of $\mathcal{P}_{\mathcal{R}}$.

(\Leftarrow) Assume the antecedent to be the case, and note that, on grounds of Formula 6, a single $h \in ABD$ is in the answer set S . Because S is not ruled out on grounds of Formula 8, it holds that this constraint does not apply. Thus, if $O = \{\mathbf{s}(\alpha_1, 1), \dots, \mathbf{s}(\alpha_m, m)\}$, then on grounds of Formula 5 holds $\exists n\{\mathbf{o}(\alpha_1, 1), \dots, \mathbf{o}(\alpha_n, n)\}n : - h. \in \mathcal{P}_{\mathcal{R}}$ for some $m \leq n$. If that is the case, then $(h \rightarrow \mathbf{o}(\alpha_1, 1), \dots, \mathbf{o}(\alpha_n, n)) \in \Theta_{\mathcal{R}}$ must hold, and $\Theta_{\mathcal{R}}, \omega(O) \approx \{h\}$. \square

COROLLARY 1. Given the conditions of Theorem 3,

$$\forall H \subseteq ABD \forall \phi, \phi' \in \mathcal{L}_0 :$$

$$(\Theta_{\mathcal{R}}, \omega(O) \approx H \ \& \ \Theta_{\mathcal{R}} \cup H \models \mathbf{g}(\tau(\phi)) \wedge \mathbf{b}(\tau(\phi'))) \iff$$

$$\mathcal{P}_{\mathcal{R}} \text{ has an answer set } S, \text{ s.t. } S \models \mathbf{g}(\tau(\phi)) \wedge \mathbf{b}(\tau(\phi'))$$

PROOF SKETCH. Observe that H entails a single instance of $\mathbf{b}/1$ and $\mathbf{g}/1$ in both $\Theta_{\mathcal{R}}$ and S (also cf. Theorem 3). \square

COROLLARY 2.

$$\forall \delta \in \mathcal{L}_{\Delta} \forall \gamma, \beta \in \mathcal{L}_0 : \quad \exists (\gamma, \beta) \in \mathbf{msa}_{\mathcal{R}}(\delta) \quad \iff$$

$$\mathcal{P}_{\mathcal{R}} \text{ has an answer set } S, \text{ s.t. } S \models \mathbf{g}(\tau(\gamma)) \wedge \mathbf{b}(\tau(\beta))$$

PROOF. From Theorems 2 and 3, and Corollary 1. \square

5. ASCRIPTION OF MENTAL STATES

This section builds upon previous sections, focusing on ascription of particular goals and beliefs.

5.1 Ascription of Rule Preconditions

In Section 3.2 the function τ was defined to translate APL terms — consisting of atoms composed by means of conjunction, disjunction, or negation — to a logical theory. Terms were translated to a functional representation, and given as arguments to predicates $\mathbf{g}/1$ and $\mathbf{b}/1$. The truth conditions of those connectives are lost in this translation, but they can be regained by decomposition of the functional arguments to constants, as shown below. It should hereby be noted that the LPARSE grounder requires particular syntactic conditions to hold with respect to the program, such that recursion cannot be used the same way as in, say, Prolog; for our implementation this means that decomposition must be specified for each level of nesting. This is not problematic, though, since the translation derives from some fixed set of APL rules for which the maximum level of nesting is known. Below are the ASP rules for decomposition of $\mathbf{conj}/2$ and $\mathbf{disj}/2$ based on the semantics of \wedge and \vee , as well as elimination of double negation and DeMorgan's laws. It is here assumed that the predicate $\mathbf{form}/1$ denotes that its argument is a valid formula, and it should be noted that the definition below is given for $\mathbf{b}/1$ but is likewise for $\mathbf{g}/1$.

$$\begin{aligned} &1\{\mathbf{b}(F1), \mathbf{b}(F2)\}2 : - \mathbf{form}(F1; F2), \mathbf{b}(\mathbf{disj}(F1, F2)). \\ &2\{\mathbf{b}(F1), \mathbf{b}(F2)\}2 : - \mathbf{form}(F1; F2), \mathbf{b}(\mathbf{conj}(F1, F2)). \\ &\mathbf{b}(F) : - \mathbf{form}(F), \mathbf{b}(\mathbf{neg}(\mathbf{neg}(F))). \\ &\mathbf{b}(\mathbf{disj}(\mathbf{neg}(F1), \mathbf{neg}(F2))) : - \\ &\quad \mathbf{form}(F1; F2), \mathbf{b}(\mathbf{neg}(\mathbf{conj}(F1, F2))). \\ &\mathbf{b}(\mathbf{conj}(\mathbf{neg}(F1), \mathbf{neg}(F2))) : - \\ &\quad \mathbf{form}(F1; F2), \mathbf{b}(\mathbf{neg}(\mathbf{disj}(F1, F2))). \end{aligned} \tag{9}$$

As expected, in case of conjunction of two formulae, both conjuncts should be satisfied. The logical connective \vee means that either or both of two disjuncts should be satisfied; this interpretation is adhered to here in decomposition of \mathbf{disj} . If a criterion of minimality is maintained then it could alternatively be stated that either single disjunct must be satisfied in order for the disjunction to be satisfied (i.e. $1\{\dots\}1$).

It is recognized in the literature that a drawback of ASP is that it is not well suited for reasoning with complex logical formulae [12]. The approach sketched above is therefore not computationally efficient if APL terms are complex, but because having single instances of $\mathbf{b}/1$ and $\mathbf{g}/1$ that point to particular β and γ does allow for straightforward proofs in preceding sections, this approach is preferred for technical simplicity. An alternative approach, left unexplored here because of lack of space, is to perform translation of compound formulae outside of ASP by bringing APL terms to a normal form and translating this directly to ASP representation.

5.1.1 Negation

In translation of negated atoms it should be noted that answer set programming offers two kinds of negation: default negation and strong (or classical) negation [12]. In principle, both can be used for interpretation of the function symbol \mathbf{neg} where it pertains to constants. The informal semantics of default negation of p are that $\mathbf{not} p$ is the case in absence of the atom p , whereas strong negation of p holds true only if the literal $\neg p$ is in the answer set. Because presence of $\mathbf{b}(\mathbf{neg}(F))$ or $\mathbf{g}(\mathbf{neg}(F))$, for some constant F , in the answer set occurs on grounds of evidence (observed actions), the use of strong negation is warranted for expressing that F is known not to be in the agent's belief/goal base; even if APL utilizes negation-as-failure (cf. [9]). Since ultimately our interest is in facts which are, or are not, ascribed to the agent as goals or beliefs, the predicates $\mathbf{bel}/1$ and $\mathbf{goal}/1$ are introduced which in contrast to $\mathbf{b}/1$ and $\mathbf{g}/1$ accept only fluents as argument (the term 'fluent' is preferred over 'constant' because valuation is fixed using these predicates).

$$\begin{aligned} &\mathbf{bel}(F) : - \mathbf{fluent}(F), \mathbf{b}(F). \\ &-\mathbf{bel}(F) : - \mathbf{fluent}(F), \mathbf{b}(\mathbf{neg}(F)). \end{aligned} \tag{10}$$

Similarly, $\mathbf{goal}/1$ is defined in relation to $\mathbf{g}/1$. Thus, observed actions warrant ascription of some fact not being the agent's goal or belief, opposed to warranting mere absence of ascription that it is. The predicates $\mathbf{bel}/1$ and $\mathbf{goal}/1$ denote fluents being (or, if negated, being not) the observed agent's belief or goal, respectively.

5.2 Plan-Based Ascription

The function CS was mentioned in Section 3.2, and defined to generate computation sequences and preserve test actions (as opposed to OS). Knowledge of successful test actions gives information about the agent's mental state, which can be informally characterized as follows: if the agent is presumed to have applied some rule and is observed to perform some action, then if this observed action is preceded by a test action, this test must have been successful.

Incorporating presumption of successful test actions into ascription requires a notion of dynamics, because tests are performed in a state which may change as a result of actions. Consider, by means of example, a plan that states that if a certain device is believed to be on, then it should be switched off. The test which succeeds before the action of

switching off the device might not succeed after performing the action. For this reason a *state argument* is added to the predicates $\mathbf{b}/1$ and $\mathbf{g}/1$. It should be noted in this regard that all occurrences of $\mathbf{b}/1$ and $\mathbf{g}/1$ in Section 4 pertain to the abduced *preconditions* of the observed agent’s rule, such that those hold in the state preceding all observed actions. This state is given the number 0, such that all $\mathbf{b}(F)$ and $\mathbf{g}(F)$ in Section 4 are extended to $\mathbf{b}(F, 0)$ and $\mathbf{g}(F, 0)$.

The expressions in Formula 9 and 10 describe general relations, such that occurrences of $\mathbf{b}/1$ and $\mathbf{g}/1$ (and, correspondingly, of $\mathbf{bel}/1$ and $\mathbf{goal}/1$) in those expressions can simply be extended with a state variable S , in regard to which the predicate $\mathbf{st}/1$ is defined to refer to valid states. Instances of this predicate can be derived from $\mathbf{o}/2$ by means of rules, or specified as facts. For example, in Formula 10 $\mathbf{bel}(F) : - \mathbf{fluent}(F), \mathbf{b}(F)$ then becomes

$$\mathbf{bel}(F, S) : - \mathbf{fluent}(F), \mathbf{st}(S), \mathbf{b}(F, S).$$

The above formulation employing a state argument is strongly reminiscent of the *situation calculus* [20], the main difference being that $\mathbf{bel}/2$ and $\mathbf{goal}/2$ refer to fluents ascribed as belief or goal to an observed agent, whereas in approaches based on the situation calculus the predicate $\mathbf{holds}/2$ refers to an agent’s own beliefs about its environment.

Ascription on grounds of computation sequences is then implemented as in Formula 11, which captures the informal notion mentioned in the first paragraph of this section and extends it to observed sequences.

$$\begin{aligned} & \forall (n : \gamma \leftarrow \beta \mid \pi) \in \mathcal{R} \\ & \forall \alpha_1, \dots, \alpha_n, \phi_1?, \dots, \phi_k?, \pi' \in \mathcal{L}_\Pi : \\ & [(\pi' \in CS(\pi) \ \& \ OS(\pi') = \{\alpha_1 \dots \alpha_i \dots \alpha_n\} \ \& \\ & \quad \pi' = \dots ; \alpha_{i-1}; \phi_1?; \dots ; \phi_k?; \alpha_i; \dots) \implies \\ & \quad (k\{\tau'(\phi_1, i-1), \dots, \tau'(\phi_k, i-1)\}k : - \\ & \quad \mathbf{r}(n, \iota(\pi')), \mathbf{s}(\alpha_1, 1), \dots, \mathbf{s}(\alpha_i, i).) \in \mathcal{P}_\mathcal{R}] \end{aligned} \quad (11)$$

The function τ' is required because plans, as defined in Section 2.1, contain test actions on expressions $\phi \in \mathcal{L}_0$ prefixed by either \mathbf{B} or \mathbf{G} . A test action $\mathbf{B}\phi?$ is evaluated with respect to the agent’s beliefs and $\mathbf{G}\phi?$ with respect to its goals, and such tests can be understood as belief/goal introspection. Translation through τ' is therefore as follows.

$$\tau'(\mathbf{B}\phi, n) = \mathbf{b}(\tau(\phi), n) \quad \tau'(\mathbf{G}\phi, n) = \mathbf{g}(\tau(\phi), n)$$

This approach has the following useful property.

PROPOSITION 1 (PROVEN FOR \mathbf{B} , LIKEWISE FOR \mathbf{G} .)
Computation sequences with mutually inconsistent tests on conjoined literals give rise to inconsistent answer sets.

PROOF. Given $\text{Lit} = \{p, \neg p \mid p \in \text{Atom}\}$, let $\phi = \bigwedge \Phi, \psi = \bigwedge \Psi$ for some $\Phi, \Psi \subseteq \text{Lit}$. Furthermore let $(n : \gamma \leftarrow \beta \mid \pi) \in \mathcal{R}$ be such that $\pi' = \dots ; \alpha_{i-1}; \mathbf{B}\phi?; \dots ; \mathbf{B}\psi?; \alpha_i; \dots \in CS(\pi)$, where $\iota(\pi') = c$. Assume that $\{\phi, \psi\} \models \perp$, such that for some $q \in \text{Lit}$ holds $\phi \rightarrow q$ and $\psi \rightarrow \neg q$, and observe that on grounds of Formulae 9, 10, and 11 it holds that every answer set satisfying $\mathbf{r}(n, c)$ and $\mathbf{s}(\alpha_1, 1), \dots, \mathbf{s}(\alpha_i, i)$ must satisfy $\mathbf{bel}(q, i-1)$ and $\neg \mathbf{bel}(q, i-1)$, and be inconsistent. \square

Proposition 1 states that the observer program which incorporates inference on grounds of presumed test actions, along the lines of Formula 11, reflects the fact that observed actions could have not been generated by a sequence which has unsatisfiable tests preceding those actions.

6. EXAMPLE

Throughout this paper short examples have been used to illustrate certain sections. In this section a more elaborate example is given, which has been implemented based on the approach presented in this paper using the grounder LPARSE [22] and the solver CLASP [11]. The example implements a (fictional) virtual character from a life-simulation game like THE SIMS that has the following two rules, in the APL syntax of Section 2.1.

```
seen_movie <- playing_movie |
  enter_mall; if B(not have_cash) then withdraw_cash
  else skip; watch_movie
have_book <- good_book or not playing_movie |
  enter_mall; if B(not have_cash) then withdraw_cash
  else skip; buy_book
```

Based on the approach sketched in this paper, those rules give rise to the following fragment of the answer set program $\mathcal{P}_\mathcal{R}$ of the observer in relation to Formulae 5, 6, and 8; abbreviations should be evident.

$$\begin{aligned} & 2\{\mathbf{g}(\mathbf{s_m}, 0), \mathbf{b}(\mathbf{p_m}, 0)\}2 : - \mathbf{r}(1, 1). \\ & 2\{\mathbf{g}(\mathbf{s_m}, 0), \mathbf{b}(\mathbf{p_m}, 0)\}2 : - \mathbf{r}(1, 2). \\ & 3\{\mathbf{o}(\mathbf{e_m}, 1), \mathbf{o}(\mathbf{w_c}, 2), \mathbf{o}(\mathbf{w_m}, 3)\}3 : - \mathbf{r}(1, 1). \\ & 2\{\mathbf{o}(\mathbf{e_m}, 1), \mathbf{o}(\mathbf{w_m}, 2)\}2 : - \mathbf{r}(1, 2). \\ & 2\{\mathbf{g}(\mathbf{h_b}, 0), \mathbf{b}(\mathbf{disj}(\mathbf{g_b}, \mathbf{neg}(\mathbf{p_m})), 0)\}2 : - \mathbf{r}(2, 3). \\ & 2\{\mathbf{g}(\mathbf{h_b}, 0), \mathbf{b}(\mathbf{disj}(\mathbf{g_b}, \mathbf{neg}(\mathbf{p_m})), 0)\}2 : - \mathbf{r}(2, 4). \\ & 3\{\mathbf{o}(\mathbf{e_m}, 1), \mathbf{o}(\mathbf{w_c}, 2), \mathbf{o}(\mathbf{b_b}, 3)\}3 : - \mathbf{r}(2, 3). \\ & 2\{\mathbf{o}(\mathbf{e_m}, 1), \mathbf{o}(\mathbf{b_b}, 2)\}2 : - \mathbf{r}(2, 4). \\ & 1\{\mathbf{r}(1, 1), \mathbf{r}(1, 2), \mathbf{r}(2, 3), \mathbf{r}(2, 4)\}1. \\ & : - \mathbf{s}(A, T), \mathbf{not} \ \mathbf{o}(A, T). \end{aligned}$$

Based on Formula 11, the rest of the program is as follows.

$$\begin{aligned} & 1\{\mathbf{b}(\mathbf{neg}(\mathbf{h_c}), 1)\}1 : - \mathbf{r}(1, 1), \mathbf{s}(\mathbf{e_m}, 1), \mathbf{s}(\mathbf{w_c}, 2). \\ & 1\{\mathbf{b}(\mathbf{h_c}), 1\}1 : - \mathbf{r}(1, 2), \mathbf{s}(\mathbf{e_m}, 1), \mathbf{s}(\mathbf{w_m}, 2). \\ & 1\{\mathbf{b}(\mathbf{neg}(\mathbf{h_c}), 1)\}1 : - \mathbf{r}(2, 3), \mathbf{s}(\mathbf{e_m}, 1), \mathbf{s}(\mathbf{w_c}, 2). \\ & 1\{\mathbf{b}(\mathbf{h_c}), 1\}1 : - \mathbf{r}(2, 4), \mathbf{s}(\mathbf{e_m}, 1), \mathbf{s}(\mathbf{b_b}, 2). \end{aligned}$$

Let $\mathcal{P}' = \mathcal{P} \cup \{\mathbf{s}(\mathbf{e_m}, 1)\}$. Focusing on the predicates $\mathbf{bel}/2$ and $\mathbf{goal}/2$, note that the program \mathcal{P}' has answer sets with $S = \{\mathbf{goal}(\mathbf{s_m}, 0), \mathbf{bel}(\mathbf{p_m}, 0)\}$ on grounds of explanations $\mathbf{r}(1, 1)$ and $\mathbf{r}(1, 2)$, and $S' = \{\mathbf{goal}(\mathbf{h_b}, 0), \mathbf{bel}(\mathbf{g_b}, 0)\}$, $S'' = \{\mathbf{goal}(\mathbf{h_b}, 0), \neg \mathbf{bel}(\mathbf{p_m}, 0)\}$, as well as $S' \cup S''$ on grounds of $\mathbf{r}(2, 3)$ and $\mathbf{r}(2, 4)$. Now let $\mathcal{P}'' = \mathcal{P}' \cup \{\mathbf{s}(\mathbf{w_m}, 2)\}$ and observe that $\mathcal{P}'' \models \mathbf{goal}(\mathbf{s_m}, 0) \wedge \mathbf{bel}(\mathbf{p_m}, 0) \wedge \mathbf{bel}(\mathbf{h_c}, 1)$ because $\mathcal{P}'' \models \mathbf{r}(1, 2)$. The example thus illustrates the theory of preceding sections in relation to specific atoms being ascribed as (not) the agent’s belief/goal, and shows conclusiveness of the observer after observation of two actions.

7. RELATED WORK

The approach presented in this paper can be categorized in the area of plan/intention recognition. Although this traditionally has been an active area of A.I. research, there has been little work *specific to agent programming*. An exception is that of Goultiaeva & Lespérance [13], who present a formal model meant for inclusion in the ConGolog programming language, which is based on the situation calculus. The approach focuses on the procedural aspect of incrementally matching observed behavior to an annotated library of plans.

An important difference with the work in this paper is the fact that our approach uses a general logical abstraction of plans and it is therefore not restricted to a particular agent programming language, except for the fact that it supposes the availability of an ASP interpreter. Furthermore, our work utilizes a mature automated (nonmonotonic) reasoning paradigm, and, last but not least, it is difficult to see how the work of Goultiaeva & Lespérance could extend to incomplete observation (i.e. missing actions), whereas our work is founded on an approach that handles this case. Of further interest is the work of Baral et al. [4] because it uses ASP to model agents knowledge about others' knowledge, albeit using a radically different approach. Apart from logic-based approaches such as the above, there exists a multitude of statistical approaches (see [7]), including some in the game domain. Such approaches require significant amounts of runtime data as input, though, and are not BDI-specific.

8. CONCLUSION AND FUTURE RESEARCH

This paper presents an answer set programming implementation of mental state ascription based on observed primitive actions of a BDI-based agent. The approach is based on earlier work ([21], cf. Section 2.2), which is reformulated here in terms of abduction in classical logic in Section 3. This logical theory gives rise to an implementation in Section 4 which utilizes the suitability of ASP for nonmonotonic reasoning, and is expanded in Section 5 to incorporate ascription based on test actions the observed agent can be presumed to have performed, hereby employing concepts of the situation calculus. Formal proof is given of useful and interesting properties of our approach.

It should be noted that for space and simplicity the implementation presented here assumes complete observation, meaning that all of the agent's actions are observed. In [21] we also formalized *incomplete observation* by means of additional structural relations which allow 'gaps' to occur in matching observed to observable sequences. By implementing those relations our implementation can be generalized to cases of incomplete observation as well. Future research can furthermore focus on exploring the relation with the situation calculus, as indicated in Section 5.2, which would furthermore open up possibilities for implementation of an agent that reasons about its own environment and mental state in relation to that ascribed to others. Existing work on ASP for dynamic domains can then be of use [12]. Furthermore, prediction of agents' actions can be considered, and, last but not least, investigating the practice of integrating logic programming interpreters into demanding applications such as games is mandatory for (industrial) deployment.

9. REFERENCES

- [1] N. Afonso and R. Prada. Agents that relate: Improving the social believability of non-player characters in role-playing games. In *Proc. of 7th Conf. on Entertainment Comp. (ICEC)*, pages 34–45, 2008.
- [2] A. Aliseda-Llera. *Seeking Explanations: Abduction in Logic, Philosophy of Science, and Artificial Intelligence*. PhD thesis, Univ. of Amsterdam, 1997.
- [3] J. Baeten and W. Weijland. *Process Algebra*. Cambridge University Press, 1990.
- [4] C. Baral, G. Gelfond, E. Pontelli, and T. C. Son. Using answer set programming to model multi-agent scenarios involving agents' knowledge about others' knowledge. In *Proceedings of 9th Intl. Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 259–266, 2010.
- [5] R. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors. *Multi-Agent Programming: Languages, Tools and Applications*. Springer, 2009.
- [6] R. I. Brafman and M. Tenenholz. Belief ascription and mental-level modelling. In *Proceedings of the 4th Intl. Conf. on Principles of Knowledge Representation and Reasoning (KR)*, pages 87–98, 1994.
- [7] S. Carberry. Techniques for plan recognition. *User Modeling & User-Ad. Interaction*, 11(1-2):31–48, 2001.
- [8] P. R. Cohen and H. J. Levesque. Intention is choice with commitment. *Artificial Intelligence*, 42(2-3):213–261, 1990.
- [9] M. Dastani. 2APL: A practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16:214–248, 2008.
- [10] D. Dennett. *The Intentional Stance*. MIT Press, 1987.
- [11] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. CLASP: A conflict-driven answer set solver. In *Proc. of the Ninth Intl. Conf. on Logic Programming and Nonmonotonic Reasoning (LPNMR)*, pages 260–265, 2007.
- [12] M. Gelfond. Answer sets. In F. van Harmelen, V. Lifschitz, and B. Porter, editors, *Handbook of Knowledge Representation*, pages 285–316. Elsevier Science, 2008.
- [13] A. Goultiaeva and Y. Lespérance. Incremental plan recognition in an agent programming framework. *Proc. of Plan, Activity and Intent Recognition (PAIR)*, 2007.
- [14] D. Harel, D. Kozen, and J. Tiuryn. *Dynamic Logic*. MIT Press, 2000.
- [15] A. C. Kakas, R. A. Kowalski, and F. Toni. *Handbook of Logic in Artificial Intelligence and Logic Programming*, volume 5, chapter The Role of Abduction in Logic Programming, pages 235–324. Oxford University Press, 1998.
- [16] J. Laird. It knows what you're going to do: Adding anticipation to a Quakebot. In *Proc. of 5th Intl. Conf. on Autonomous Agents*, pages 385–392, 2001.
- [17] A. B. Loyall. *Believable Agents*. PhD thesis, Carnegie Mellon University, 1997.
- [18] P. Novák. Jazzyk: A programming language for hybrid agents with heterogeneous knowledge representations. In *Proceedings of ProMAS 2008*, pages 72–87, 2009.
- [19] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proc. of the 2nd Intl. Knowl. Repr. Conf. (KR)*, pages 473–484, 1991.
- [20] R. Reiter. The frame problem in the situation calculus: A simple solution (sometimes) and a completeness result for goal regression. *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, pages 359–380, 1991.
- [21] M. P. Sindlar, M. Dastani, F. Dignum, and J.-J.Ch. Meyer. Mental state abduction of BDI-based agents. In *Proc. of the 6th Intl. Workshop on Declarative Agent Lang. and Tech. (DALT)*, pages 161–178, 2008.
- [22] T. Syrjänen. LPARSE manual. (<http://www.tcs.hut.fi/Software/smodels/>).

Social Choice Theory

Possible and necessary winners in voting trees: majority graphs vs. profiles

Maria Silvia Pini
University of Padova
Padova, Italy
mpini@math.unipd.it

Francesca Rossi
University of Padova
Padova, Italy
frossi@math.unipd.it

Kristen Brent Venable
Universita' di Padova
Padova, Italy
kvenable@math.unipd.it

Toby Walsh
NICTA and UNSW
Sydney, Australia
toby.walsh@nicta.com.au

ABSTRACT

Given the preferences of several agents over a common set of candidates, voting trees can be used to select a candidate (the winner) by a sequence of pairwise competitions modelled by a binary tree (the agenda). The majority graph compactly represents the preferences of the agents and provides enough information to compute the winner. When some preferences are missing, there are various notions of winners, such as the possible winners (that is, winners in at least one completion) or the necessary winners (that is, winners in all completions). In this generalized scenario, we show that using the majority graph to compute winners is not correct, since it may declare as winners candidates that are not so. Nonetheless, the majority graph can be used to compute efficiently an upper or lower approximation of the correct set of winners.

Categories and Subject Descriptors

I.2.11 [Computing methodologies]: Artificial Intelligence—*Distributed Artificial Intelligence*

General Terms

Algorithms, Theory

Keywords

Preferences, incompleteness, necessary winners, voting trees

1. INTRODUCTION

Voting is a simple and natural mechanism to aggregate the preferences of multiple agents. Results like those of Gibbard and Satterthwaite demonstrate that, under weak assumptions like an election of more than two candidates, no voting rule is ideal. Many different voting rules, with different properties, have therefore been proposed. Voting trees are a general method that can implement many such rules [2]. A voting tree is a binary tree (called the agenda)

Cite as: Possible and necessary winners in voting trees: majority graphs vs. profiles, Maria Silvia Pini, Francesca Rossi, Kristen Brent Venable, and Toby Walsh, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 311-318. Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

where the leaves are labelled with the candidates in the election, the internal nodes are labelled with the winner of the pairwise comparison between the children, and the root wins the overall election. Voting trees can implement, for example, any voting rule over three candidates [3]. Voting trees have therefore been studied as a general, abstract model for decision-making among multiple agents (see, for example, [10, 9, 11]).

In practice, we often want to make decisions despite the presence of uncertainty. Uncertainty can come in different forms. There may be uncertainty about the votes [4, 12, 1, 13]. We may, for example, have only partially elicited preferences, or we may only have partial knowledge about the votes of other agents. There may also be uncertainty about the voting rule itself. For instance, with voting trees, there may be uncertainty about the agenda [7]. This may be because the agenda is not yet fixed (for example, the agenda is to be chosen at a later date by means of a random draw as in the World Cup), or the agenda is not announced in advance (to impede manipulation), or the chair may still be in a position to change the agenda (in order to manipulate the result). We are therefore interested in computing the results of an election with a voting tree where there is uncertainty in the vote and/or in the voting tree.

When some preferences are missing, there are various notions of winners, such as the possible winners (that is, winners in at least one completion) or the necessary winners (that is, winners in all completions) [4]. When computing possible and necessary winners, we can work from the incomplete votes or the majority graph. The majority graph summarizes the votes in the form of a directed graph where there is a directed arc between two candidates iff a majority of the agents prefer the first candidate over the second. We can also consider whether the agenda is fixed or unknown. Previously, Lang et al. [5] studied the problem of computing possible and necessary winners from the majority graph. In [8], Pini et al. compared these results to the problem of computing possible and necessary winners from incomplete profiles. However, this work left open several important relationships between the different types of winners, which we close here.

To be precise, we consider all notions of winners in the setting of incomplete preferences or uncertain agenda (possible Condorcet, necessary Condorcet, possible Schwartz, necessary Schwartz, possible, and necessary winners) and study

whether each of these sets of winners is computable by just looking at the majority graph, without having to consider the whole profile. In other words, we study if the set of winners computed from the majority graph coincides with that computed from the profile. The reason to do this is that computing winners from the majority graph takes polynomial time, even when there are exponential many completions.

For some notions of winners, the literature already tells us if the two sets coincide or not [8]. In fact, we know that, when considering simple voting trees (that is, agendas where each candidate appears exactly once as a leaf), possible Condorcet winners and necessary Condorcet winners can be safely computed from the majority graph, while this is not so for possible Schwartz winners. We first close all open questions about simple voting trees, showing that unfortunately equality does not hold in general for all remaining notions of winners. We then examine the more general setting of voting trees, where we show that the results for simple voting trees are maintained. This means that, both in simple voting trees and in general voting trees, reasoning with the majority graph always produces correct results only when we look for possible and necessary Condorcet winners. When we aim to find other kinds of winners, working with the majority graph gives an upper or lower approximation of the correct set of winners. The situation is even worse for voting trees, since some polynomial algorithms that correctly return winners from the majority graph in the simple voting tree setting do not work well for voting trees, since they may return incorrect responses also w.r.t. the majority graph-based notion of winners. However, what they return is still a lower or upper approximation of the correct set of winners.

2. BASIC NOTIONS

We now give the basic notions for (simple) voting trees [5, 8, 6].

2.1 Preferences, profiles, and majority graphs

We assume that each agent's *preferences* are specified by a strict total order (TO), that is, by an asymmetric, transitive and complete order, on a set of m candidates. The candidates are taken from a set Ω , and they represent the possible options over which agents vote.

A *profile* P on Ω is a collection of n strict total orders over Ω , i.e., $P = (P_1, \dots, P_n)$, where P_i is the preference relation of agent i . An *incomplete preference relation* $>$ on Ω is a strict order on Ω , that is, a transitive and irreflexive relation on Ω . An *incomplete profile* on Ω is a collection $P = (P_1, \dots, P_n)$ of incomplete preference relations on Ω . Let $P = (P_1, \dots, P_n)$ be an incomplete profile over a set of candidates Ω , a completion R of P is a tuple (R_1, \dots, R_n) such that every R_i is a strict total order on Ω containing P_i .

For simplicity, we assume that the number of the agents is odd.

Given an (incomplete) profile P , the *majority graph* $M(P)$ induced by P is the directed graph whose set of vertices is Ω , and where an edge from A to B (denoted by $A >_m B$) denotes a strict majority of voters who prefer A to B . A majority graph is said to be complete if, for any two vertices, there is a directed edge between them, and fully incomplete if there are no edges. Also, if $M(P)$ is incomplete, the set of all complete majority graphs extending $M(P)$ corresponds

to a (possibly proper) superset of the set of complete majority graphs induced by all possible completions of P .

2.2 Voting trees

Given a set of candidates, the *simple voting tree rule* (resp., *voting tree rule*) is defined by a binary tree with one candidate per leaf. Each candidate appears exactly once in the leaves (resp., each candidate may appear more than once in the leaves). Each internal node represents the candidate that wins the pairwise election between the node's children. The winner of every pairwise election is computed by the majority rule, where A beats B iff there is a majority of votes stating $A > B$. The candidate at the root of the tree is the overall winner. Given a complete profile, candidates which win for every (simple) voting tree are called *Condorcet winners* and candidates which win for at least one (simple) voting tree are called *Schwartz winners*.

2.3 Notions of winners

Various kinds of winners have been defined from incomplete profiles and from incomplete majority graphs [5, 8, 6].

DEFINITION 1. Let P be an incomplete profile and A a candidate.

- A is a possible Schwartz winner for P (i.e., $A \in PossS(P)$) iff there exists a completion of P and a simple voting tree for which A wins;
- A is the necessary Schwartz winner for P (i.e., $A \in NecS(P)$) iff for every completion of P there is a simple voting tree for which A wins;
- A is a possible Condorcet winner for P (i.e., $A \in PossC(P)$) iff there is a completion of P such that A is a winner for every simple voting tree;
- A is the necessary Condorcet winner for P (i.e., $A \in NecC(P)$) iff for every completion of P , and for every simple voting tree, A is a winner.

When the voting tree is given, the following two notions of winners can also be considered [6, 12].

DEFINITION 2. Let P be an incomplete profile, A a candidate, and T a simple voting tree.

- A is a possible winner for P and T (i.e., $A \in Poss(P, T)$) iff there exists a completion of P for which A wins in T ;
- A is the necessary winner for P and T (i.e., $A \in Nec(P, T)$) iff, for every completion of P , A wins in T .

When the profile is complete, necessary and possible Condorcet winners coincide. The same holds also for necessary and possible Schwartz winners, and for necessary and possible winners.

The definitions of winners given above can be defined also from incomplete majority graphs [5, 8]. The only difference is such definitions consider the completions of the incomplete majority graph and not those of the incomplete profile.

2.4 Majority graph vs. profile

When the profile is complete, the possible and necessary versions of the same notion of winner collapse, and reasoning from the majority graph is enough for correctly computing the winner. However, when the profile is incomplete, considering the majority graph rather than the profile may give different results. A majority graph may have completions that do not correspond to any completion of the profile. Thus, for certain notions of winners, the possible/necessary winners for the incomplete profile do not coincide with the possible/necessary winner for the incomplete majority graph.

Of course, the correct notion is that defined from the profile, but in some cases the two notions of winners coincide, so we can safely consider only the majority graph. This is more convenient since the majority graph is a compact representation of the profile. Reasoning over this structure is therefore more efficient.

In Table 1 we summarize what is known about the relationship between the set of winners from an incomplete profile P and the set of winners from the incomplete majority graph $M(P)$, both in simple voting trees and in voting trees.

	SVT	VT
Possible Schwartz winners	\neq [8]	?
Necessary Schwartz winners	?	?
Possible Condorcet	$=$ [8]	?
Necessary Condorcet	$=$ [8]	?
Possible winners	?	?
Necessary winners	?	?

Table 1: State of the art about winners computed from majority graph or profile, for simple voting trees (SVT) and voting trees (VT).

In words, we know that possible Condorcet winners and necessary Condorcet winners can be correctly computed from the majority graph, while this is not so for possible Schwartz winners. However, we don't know anything about the other notions of winners, except the obvious subset inclusion that comes from the simple observation that the majority graph may have more completions than the profile. More precisely, every notion in its "possible" version and related to the profile denotes a subset of the same notion related to the majority graph. For example, $Poss(P, T) \subseteq Poss(M(P), T)$. Vice versa, every notion in its "necessary" version and related to the majority graph denotes a subset of the same notion related to the profile. For example, $Nec(M(P), T) \subseteq Nec(P, T)$.

2.5 Computing majority graph winners

All the sets of winners for the majority graph can be computed in polynomial time for simple voting trees [5, 6].

Given a simple voting tree T and an incomplete majority graph G , algorithm *Win*, presented in [6], computes the set of possible winners for G and T . This algorithm (see the pseudocode below) recursively takes in input a simple voting tree T , an incomplete majority graph G , and it returns a set of candidates W , which is the set of possible winners for G and T . If $root(T)$ is not empty, and both $left(T)$ (i.e., the left subtree of T) and $right(T)$ (i.e., the right subtree of T) are empty, then the algorithm returns $label(root(T))$ (i.e., the candidate which labels the root of T). Otherwise, the set of winners at the root of T is the set of all candidates

who are possible winners in the left (resp., right) branch of T and who beat at least one candidate who is a possible winner in the right (resp., left) branch of T .

Algorithm *StrongWin* [6] runs algorithm *Win* on T and G , and just checks if the output is a single candidate. If so, it declares it the necessary winner, otherwise it returns the empty set as there is then no necessary winner.

Algorithm 1: *Win*

```

Input:  $T$ : a simple voting tree,  $G$ : an incomplete majority graph;
Output:  $W$ : set of candidates;
if  $T$  contains only one node then
   $W \leftarrow label(root(T))$ 
else
   $W_1 \leftarrow Win(left(T), G)$ ;
   $W_2 \leftarrow Win(right(T), G)$ ;
   $W \leftarrow \emptyset$ ;
  foreach  $(s, t) \in W_1 \times W_2$  do
    if  $s >_m t$  then
       $W \leftarrow W \cup \{s\}$ 
    else
      if  $t >_m s$  then
         $W \leftarrow W \cup \{t\}$ 
      else
         $W \leftarrow W \cup \{s, t\}$ 
  return  $W$ 

```

3. WINNERS IN SIMPLE VOTING TREES

We now show that the set of the necessary Schwartz winners (resp., possible winners, necessary winners) for an incomplete profile P may be different from the set of the necessary Schwartz winners (resp., possible winners, necessary winners) for the incomplete majority graph $M(P)$ in simple voting trees. These results close all the open questions in the simple voting tree column of Table 1.

3.1 Necessary Schwartz winners

As noted above, a necessary Schwartz winner from an incomplete majority graph $M(P)$ is always a necessary Schwartz winner from the incomplete profile P [8]. More precisely, let P be an incomplete profile. Then $NecS(M(P)) \subseteq NecS(P)$. However, in general, the opposite does not hold.

THEOREM 1. *There is an incomplete profile P such that $NecS(P) \neq NecS(M(P))$.*

PROOF. To show the result, we give an incomplete profile P and a candidate A such that $A \in NecS(P)$ and $A \notin NecS(M(P))$.

Let $\Omega = \{A, A_1, B_1, B_2, B_3\}$. Assume that we have 5 agents and that the incomplete profile P is defined as follows:

- agent 1: $(A_1 > B_2 > B_3, A > B_1)$;
- agent 2: $(B_2 > B_3 > A_1 > B_1 > A)$;
- agent 3: $(A > A_1 > B_3 > B_1 > B_2)$;
- agent 4: $(B_1 > A > B_2 > B_3 > A_1)$;
- agent 5: $(B_3 > B_1 > B_2 > A > A_1)$.

Given this profile, the corresponding majority graph has the following edges:

- $A >_m A_1, B_1 >_m A, B_1 >_m B_2,$
- $B_2 >_m B_3, B_2 >_m A_1, B_3 >_m B_1, B_3 >_m A_1.$

By Theorem 8 of [5], which states that $A \in NecS(M(P))$ iff there is a path from A to every other candidate in $M(P)$, since there is no path from A to B_1 in $M(P)$, then $A \notin NecS(M(P))$. However, $A \in NecS(P)$. In fact, for every completion of P , there is a tree where A wins. Notice that in P only the first agent expresses incomplete preferences. Therefore, the completions of P are as many as the completions of the first agent's preferences. Such completions can be partitioned in two types: those where the first agent puts A_1 above A and those where the first agent puts A above A_1 :

- In every completion of P where A_1 is above A in the first agent's preferences, we have, by transitivity, $A_1 > B_1$ for this agent. This yields a majority of agents stating $A_1 > B_1$. Therefore, for every completion of this kind, the corresponding majority graph has the edge $A_1 >_m B_1$. It is possible to see that A wins in the tree where first B_2 plays against B_3 , the winner (that is, B_2) plays against B_1 , the winner (that is, B_1) plays against A_1 , and finally the winner (that is, A_1) plays against A .
- In every completion of P where A is above A_1 in the first agent's preferences, we have, by transitivity, $A > B_2$ for this agent. This yields a majority of agents stating $A > B_2$. Therefore, for every completion of this kind, the corresponding majority graph has the edge $A >_m B_2$. It is possible to see that A wins in the tree where first B_3 plays against B_1 , the winner (that is, B_3) plays against B_2 , the winner (that is, B_2) plays against A , and finally the winner (that is, A) plays against A_1 . \square

However, in some restricted cases the two notions of winners coincide. For example, when we have 3 candidates, the necessary Schwartz winners from the incomplete profile and from the incomplete majority graph coincide.

THEOREM 2. *Let P be an incomplete profile over 3 candidates. Then $NecS(P) = NecS(M(P))$.*

PROOF. For every candidate A , we can partition the set of candidates in two sets S_1 and S_2 , where S_1 contains the candidates that are reachable from A (i.e., every candidate X such that there is a path $A >_m \dots >_m X$ from A to X in $M(P)$) and S_2 contains the candidates that are not reachable from A . If there are 3 candidates, (say A, B , and C), then there are four possible kinds of majority graphs, depending on who reaches who else:

- $S_1 = \{A\}$ and $S_2 = \{B, C\}$;
- $S_1 = \{A, B\}$ and $S_2 = \{C\}$;
- $S_1 = \{A, C\}$ and $S_2 = \{B\}$;
- $S_1 = \{A, B, C\}$ and $S_2 = \emptyset$;

In the first case, there is no path from A to B . Therefore, by Theorem 8 of [5], $A \notin NecS(M(P))$. Moreover, B or C has no ingoing edges in $M(P)$. Assume that B has no ingoing edges. Let us consider the completion of P , say P' ,

where we put $B > A$ if the relation between A and B is unspecified, and $B > C$ if the relation between B and C is unspecified. Then, B has only outgoing edges in $M(P')$ and so B is a Condorcet winner, i.e., he wins in every tree. Since there is a completion of P where A is a loser for every tree, $A \notin NecS(P)$.

In the second case, there is no path from A to C . Therefore, by Theorem 8 of [5], $A \notin NecS(M(P))$. Moreover, C has no ingoing edges in $M(P)$. Let us consider the completion of P , say P' , where we put $C > A$ if the relation between A and C is unspecified, and $C > B$ if the relation between B and C is unspecified. Then, C has only outgoing edges in $M(P')$ and so B is a Condorcet winner, i.e., he wins in every tree. Since there is a completion of P where A is a loser for every tree, $A \notin NecS(P)$.

In the third case, there is no path from A to B . Therefore, by Theorem 8 of [5], $A \notin NecS(M(P))$. Moreover, B has no ingoing edges in $M(P)$. We can conclude that $A \notin NecS(P)$ via a reasoning that is similar to the one used in the case above.

In the fourth case, there is a path from A to every other candidate. Therefore, $A \in NecS(M(P))$ and so $A \in NecS(P)$ since $NecS(M(P)) \subseteq NecS(P)$. \square

The equality between $NecS(P)$ and $NecS(M(P))$ holds also when we have more than 3 candidates, if we impose some other restrictions.

THEOREM 3. *Let P be an incomplete profile. Then $NecS(P) = NecS(M(P))$ if*

- $M(P)$ is complete, or
- $M(P)$ is fully incomplete, or
- there are two candidates with no ingoing edges in $M(P)$ (in which case $NecS(P) = NecS(M(P)) = \emptyset$).

PROOF.

- If $M(P)$ is complete, then $M(P)$ is also the majority graph of every completion of P . Therefore, if $A \notin NecS(M(P))$, there is no path from A to some candidate B in $M(P)$ and in $M(P')$, for every completion P' of P . Therefore, $A \notin NecS(P)$ and so $NecS(P) \subseteq NecS(M(P))$. We can thus conclude that $NecS(P) = NecS(M(P))$, since [8] shows that $NecS(P) \supseteq NecS(M(P))$.
- If $M(P)$ is fully incomplete, then $NecS(M(P)) = \emptyset$. Moreover, if $M(P)$ is fully incomplete, there are two candidates, say B_1 and B_2 , with no ingoing edges. If we consider the completion P_1 of P where we put $B_1 > C$ for every C such that the relation between B_1 and C is unspecified, B_1 is a Condorcet winner, i.e., B_1 wins in every tree. Similarly, if consider the completion P_2 of P where we put $B_2 > C$ for every C such that the relation between B_2 and C is unspecified, B_2 is a Condorcet winner, i.e., B_2 wins in every tree. Since in P_1 B_1 wins for every tree, and since in P_2 B_2 wins for every tree, it is not possible to find a unique candidate that in both completions P_1 and P_2 wins for some tree. Therefore, $NecS(P) = \emptyset$.
- If there are two candidates with no ingoing edges in $M(P)$, then we can conclude as in the case above that $NecS(P) = NecS(M(P))$. \square

We now consider cases where a candidate is neither a necessary Schwartz winner from the majority graph nor a necessary Schwartz winner from the profile.

THEOREM 4. *Let P be an incomplete profile.*

- *If there is a unique candidate B with no ingoing edges, then, for every other candidate A , $A \notin \text{NecS}(M(P))$ and $A \notin \text{NecS}(P)$.*
- *For every candidate A , such that there is at least a candidate that A does not reach and all the candidates that are reachable from A in $M(P)$ are beaten by all the other candidates that are not reachable from A , $A \notin \text{NecS}(M(P))$ and $A \notin \text{NecS}(P)$.*
- *If there is a partition of the candidates in two sets, say S_1 and S_2 , such that every element in S_1 is worse than every element in S_2 in $M(P)$, then for every $A \in S_1$, $A \notin \text{NecS}(M(P))$ and $A \notin \text{NecS}(P)$.*

PROOF.

- If there is a unique candidate B with no ingoing edges, then, for every other candidate $A \neq B$, A does not reach B . Thus, by Theorem 8 of [5], $A \notin \text{NecS}(M(P))$. Let us consider the completion P' where we put $B > C$ for every C such that the relation between B and C is unspecified. Then B is a Condorcet winner, i.e., B wins in every tree, and so every other candidate A is a loser for every tree. Therefore $A \notin \text{NecS}(P)$.
- Let us consider a candidate A such that all the candidates that are reachable from A in $M(P)$ are beaten by all the other candidates that are not reachable from A . Let us denote with B the candidate that A does not reach. Since A does not reach B , then, by Theorem 8 of [5], $A \notin \text{NecS}(M(P))$. Let us consider the completion P_1 of P where we put C above A for every C where the relation between C and A is unspecified in $M(P)$ (and thus also $B > A$ if the relation between A and B is unspecified in $M(P)$) and where we put all the remaining unspecified preferences in an arbitrary way that satisfies transitivity. A cannot reach B in $M(P_1)$. In fact, A cannot reach B directly by construction. Moreover, A cannot reach B via the candidates that A reaches since, by hypothesis, such candidates are all beaten by every candidate that A does not reach in $M(P)$. Therefore, they are all beaten by B in $M(P)$ and thus also in $M(P_1)$. Since A cannot reach B in the complete majority graph $M(P_1)$, by Theorem 8 of [5], A is not a Schwartz winner for P_1 , i.e., for P_1 , A is a loser for every tree. Therefore, $A \notin \text{NecS}(P)$.
- If there is a partition of the candidates in two sets, say S_1 and S_2 , such that every element in S_1 is worse than every element in S_2 in $M(P)$, then we can conclude by using a reasoning similar to the one considered in the previous item. \square

3.2 Possible winners

Given an incomplete profile P and a simple voting tree T , the possible winners of T for P and for $M(P)$ may be different.

THEOREM 5. *There is an incomplete profile P and a simple voting tree T such that $\text{Poss}(P, T) \neq \text{Poss}(M(P), T)$.*

PROOF. We can consider the incomplete profile P with just one agent and $\Omega = \{A, B, C\}$, where only the relation between A and B is specified and it is $A > B$. The induced majority graph $M(P)$ has only one edge from A to B . Let us consider the simple voting tree T where A plays against C and the winner plays against B . It is easy to see that $B \in \text{Poss}(M(P), T)$, since there is a completion of $M(P)$ where B wins in T , i.e., $B >_m C >_m A$. However, for every completion of P , B does not win in T and so $B \notin \text{Poss}(P, T)$. \square

3.3 Necessary winners

In general, if there is a necessary winner from an incomplete majority graph $M(P)$, then it is also a necessary winner from the incomplete profile P . More precisely, let P be an incomplete profile and T a simple voting tree, then $\text{Nec}(M(P), T) \subseteq \text{Nec}(P, T)$.

We will now show that the opposite does not hold in general.

THEOREM 6. *There is an incomplete profile P and a simple voting tree T such that $\text{Nec}(P, T) \neq \text{Nec}(M(P), T)$.*

PROOF. To show the result, we give an incomplete profile P , a simple voting tree T , and a candidate A such that $A \in \text{Nec}(P, T)$ and $A \notin \text{Nec}(M(P), T)$.

Let $\Omega = \{A, B, C, D, E, F\}$. Assume to have 5 agents and that the incomplete profile P is defined as follows:

- agent 1: $(E > B > C, F > D > A)$;
- agent 2: $(A > E > F > D > B > C)$;
- agent 3: $(A > C > D > F > E > B)$;
- agent 4: $(C > D > F > E > B > A)$;
- agent 5: $(B > A > F > E > C > D)$.

The majority graph of P has the following edges:

- $A >_m C, A >_m D, A >_m E, A >_m F,$
- $B >_m C, B <_m D, B <_m E, B <_m F,$
- $C >_m D, C <_m E, D <_m F, E <_m F.$

Assume that the voting tree T is defined as follows: the winner between C and F plays against the winner between E and D , the winner then plays against B and finally the winner plays against A .

If we apply Algorithm *Win* (which is described in Section 2.5) to T and $M(P)$, then the returned set is $\{A, B\}$. Since T is a simple voting tree, the set $\{A, B\}$ coincides with the set of the possible winners for $M(P)$ and T . Since there are two possible winners, then $\text{Nec}(M(P), T) = \emptyset$.

However, $A \in \text{Nec}(P, T)$. In fact, for every completion of P , A wins in T . Note that in P only the first agent expresses incomplete preferences. Therefore, the completions of P correspond to the completions of the first agent's preferences. Such completions can be partitioned into two types: those where the first agent puts E above F and those where the first agent puts F above E .

- In every completion of P where E is above F in the first agent's preferences, we have, by transitivity, $E > D$ for this agent, and so there is a majority of agents stating $E > D$. Therefore, for every completion of this kind, its majority graph has the edge $E >_m D$, and thus A wins in T , since C is beaten by F or by E , then B is beaten by F and E , and finally A beats both E and F . Therefore, A is the overall winner.
- In every completion of P where F is above E in the first agent's preferences, we have, by transitivity, $F > C$ for this agent, and so there is a majority of agents stating $F > C$. Therefore, for every completion of this kind, its corresponding majority graph has the edge $F >_m C$, and thus A wins in T , since C is beaten by F , then F beats both D and E , then F beats B , and finally A beats F . Therefore, A is the overall winner. \square

However, in some restricted cases the two notions of winners coincide. For example, when we have 3 candidates, the necessary winners from the incomplete profile and from the incomplete majority graph coincide.

THEOREM 7. *Let P be an incomplete profile over 3 candidates and T a simple voting tree. Then, $Nec(P, T) = Nec(M(P), T)$.*

PROOF. When there are three candidates, say A , B , and C , there is only one kind of tree, where one candidate plays against another candidate, and the winner plays against the remaining one. Let us consider the simple voting tree T where A plays against B and the winner plays against C . Every other simple voting tree can be obtained by T by renaming the candidates. We now show that for every possible kind of incomplete majority graph $M(P)$, $Nec(P, T) = Nec(M(P), T)$. We will say $A ?_m B$ when the relation between A and B is missing in $M(P)$.

Assume $A >_m B$. If $A >_m C$ or $A <_m C$, $M(P)$ gives all the information for T and thus $Nec(M(P), T) = Nec(P, T)$. If $A ?_m C$, $Poss(M(P), T) = \{A, C\}$ and thus $Nec(M(P), T) = \emptyset$. In the completion of P where we put $A > C$ if the relation between A and C is unspecified, A wins in T , while in the completion of P where we put $C > A$, if the relation between A and C is unspecified, C wins in T . Therefore, since there is no a single candidate that in every completion of P wins in T , $Nec(P, T) = \emptyset$.

Assume $A <_m B$. We can conclude similarly to the previous item.

Assume $A ?_m B$.

- If $C >_m A$ and $C >_m B$, then $Nec(M(P), T) = Nec(P, T) = \{C\}$.
- If $C <_m A$ and $C <_m B$, then $Poss(M(P), T) = \{A, B\}$ and thus $Nec(M(P), T) = \emptyset$. In the completion of P where we put $A > B$ (resp., $B > A$) if the relation between A and B is unspecified, A (resp., B) wins in T and thus $Nec(P, T) = \emptyset$.
- If $C <_m A$ and $C >_m B$, then $Poss(M(P), T) = \{A, C\}$ and thus $Nec(M(P), T) = \emptyset$. In the completion of P where we put $A > B$ (resp., $B > A$) if the relation between A and B is unspecified, A (resp., C) wins in T and thus $Nec(P, T) = \emptyset$.

- If $C >_m A$ and $C <_m B$, we can conclude similarly to the previous item.
- If $C >_m B$ and $C ?_m A$, then $Poss(M(P), T) = \{C, A\}$ and thus $Nec(M(P), T) = \emptyset$. Moreover, in the completion of P where we put $C > A$ if the relation between A and C is unspecified, C wins in T , while in the completion of P where we put $A > B$ and $A > C$, A wins in T . Therefore, since there is no a single candidate that in every completion of P wins in T , $Nec(P, T) = \emptyset$.
- If $C >_m A$ and $C ?_m B$, then we can conclude similarly to the previous item.
- If $C <_m B$ and $C ?_m A$, then $Poss(M(P), T) = \{A, B, C\}$ and thus $Nec(M(P), T) = \emptyset$. Moreover, in the completion of P where we put $B > A$, if the relation between A and B is unspecified, B wins in T , while in the completion of P where we put $A > C$ if the relation between A and C is unspecified, and $A > B$ if the relation between A and B is unspecified, A wins in T . Therefore, since there is no single candidate that in every completion of P wins in T , $Nec(P, T) = \emptyset$.
- If $C <_m A$ and $C ?_m B$, then we can conclude similarly to the previous item. \square

The equality between $Nec(P, T)$ and $Nec(M(P), T)$ holds also when we have more than 3 candidates, if we impose some other restrictions.

THEOREM 8. *Let P be an incomplete profile and T a simple voting tree. Then $Nec(P, T) = Nec(M(P), T)$ if*

- $M(P)$ is complete, or
- $M(P)$ is fully incomplete (in which case $Nec(P, T) = Nec(M(P), T) = \emptyset$).

PROOF.

- If $M(P)$ is complete, then $M(P)$ is also the majority graph of every completion of P . Therefore, if $A \notin Nec(M(P), T)$, then $A \notin Nec(P, T)$. Therefore, $Nec(P, T) \subseteq Nec(M(P), T)$. We can thus conclude that $Nec(P, T) = Nec(M(P), T)$, since $Nec(P, T) \supseteq Nec(M(P), T)$.
- If $M(P)$ is fully incomplete, then all the candidates are possible winners for $M(P)$ and so $Nec(M(P), T) = \emptyset$. Moreover, if $M(P)$ is fully incomplete, there are two candidates, say B_1 and B_2 with no ingoing edges. We can conclude that $Nec(P, T) = \emptyset$ by using a reasoning similar to the one considered in the proof of the second item of Theorem 3. More precisely, in the completion P_1 of P where we put $B_1 > C$ for every C such that the relation between B_1 and C is unspecified, B_1 is a Condorcet winner, and so B_1 wins also in T , while in the completion P_2 of P where we put $B_2 > C$ for every C such that the relation between B_2 and C is unspecified, B_2 is a Condorcet winner, and so B_2 wins also in T . Therefore, it is not possible to find a unique candidate that in the completions P_1 and P_2 wins in T . Therefore, $Nec(P, T) = \emptyset$. \square

We can also identify cases in which a candidate is neither a necessary winner from the majority graph nor a necessary winner from the profile.

THEOREM 9. *Let P be an incomplete profile and T a simple voting tree. For every candidate A such that, for every candidate C that may play against A in T , $A <_m C$ or the relation between A and C is unspecified in $M(P)$, $A \notin Nec(M(P), T)$ and $A \notin Nec(P, T)$.*

PROOF. If, for every candidate C that may play against A in T , we have $A <_m C$, then $A \notin Poss(M(P), T)$. Thus $A \notin Nec(M(P), T)$ and $A \notin Poss(P, T)$. Since $A \notin Poss(P, T)$, we have that $A \notin Nec(P, T)$. If there is a candidate C that may play against A in T , such that the relation between A and C is unspecified in $M(P)$, then either $A \notin Poss(M(P), T)$ (and so we can conclude as above) or $A \in Poss(M(P), T)$ but $|Poss(M(P), T)| > 1$. Then, by Algorithm *StrongWin* [6], $A \notin Nec(M(P), T)$. Moreover, let us consider the completion of P where we put $A < C$ for every C such that the relation between A and C is unspecified in $M(P)$. Then, for every tree (and thus also in T) A is a loser. Therefore, $A \notin Nec(P, T)$. \square

4. WINNERS IN VOTING TREES

We now close the open questions of Table 1 in the voting tree column. We will show that algorithms *Win* and *StrongWin* [6], that correctly compute possible and necessary winners from an incomplete majority graph and a simple voting tree, are not correct any longer when we consider voting trees.

4.1 Winners

We first notice that, since a simple voting tree is a voting tree, all the inequality results shown in the previous section (or already known and shown in Table 1) for simple voting trees hold also for voting trees. Thus, we have inequalities for the notions of possible Schwartz winners, necessary Schwartz winners, possible winners, and necessary winners.

For the remaining notions of possible and necessary Condorcet winners, we will now show that the equalities that hold for simple voting trees hold also for voting trees.

THEOREM 10. *Let P be an incomplete profile. Assume we consider voting trees. Then*

- $PossCond(P) = PossCond(M(P))$;
- $NecCond(P) = NecCond(M(P))$.

PROOF. It follows from the proofs of items 3 and 4 of Theorem 1 of [8], which show that these equalities hold for simple voting trees, since this proof depends only on the completions of $M(P)$ and of P and not on the kind of voting trees considered. \square

4.2 Majority graph winners

It is easy to see that all the polynomial time algorithms presented for incomplete majority graphs and simple voting trees in [5] are sound and complete also for voting trees, since they don't exploit the fact that each candidate appears exactly once in the leaves.

However, we can show that, when we consider voting trees instead of simple voting trees, algorithms *Win* and *StrongWin* [6], that compute possible and necessary winners from an incomplete majority graph and a simple voting tree, are not correct. In fact, given a voting tree T and a majority

graph M , Algorithm *Win* may return also candidates that are not possible winners for M and T , and algorithm *StrongWin* may return the empty set even if there is a necessary winner for M and T .

THEOREM 11. *There is an incomplete majority graph M and a voting tree T such that, if W is the set of candidates returned by Algorithm *Win* [6] applied to M and T , then $W \neq Poss(M, T)$.*

PROOF. We give an incomplete majority graph M and a voting tree T , such that the set of candidates returned by Algorithm *Win* [6] applied to T and M contains also a candidate which is not a possible winner for T and M .

Assume we have 5 candidates, say A, B, C, D , and E , and that the incomplete majority graph M has the following edges: $A >_m D, A >_m E, B >_m A, B >_m E, C >_m A, C >_m D, D >_m B, E >_m C$. Assume that the voting tree T is defined as follows:

- $left(root(T))$ is the voting tree where first B plays against C , the winner plays against D , and finally the winner plays against A , and
- $right(root(T))$ is the voting tree where first B plays against C , the winner plays against E , and finally the winner plays against A .

The set of the candidates returned by Algorithm *Win* [6] is the set $\{A, B, C\}$, while the set of possible winners for M and T is the set $\{B, C\}$. A is not a possible winner for M and T , since there are only two completions of M , i.e., the one, say c_1 , where we put $B > C$ and the one, say c_2 , where we put $C > B$. In the first one, the winner is B , while in the second one the winner is C :

- In completion c_1 , A is the winner of $left(root(T))$ (since B beats C , then B is beaten by D , and finally D is beaten by A), B is the winner of $right(root(T))$ (since B beats C , then B beats E , and finally B beats by A), and thus, since $B >_m A$ in M , B is the winner of T .
- In completion c_2 , C is the winner of $left(root(T))$ (since C beats B , then C beats D , and finally C beats A), A is the winner of $right(root(T))$ (since C beats B , then C is beaten by E , and finally E is beaten by A), and thus, since $C >_m A$ in M , C is the winner of T . \square

THEOREM 12. *There is an incomplete majority graph M and a voting tree T such that, if W the set of candidates returned by Algorithm *StrongWin* [6] applied to M and T , $W \neq Nec(M, T)$.*

PROOF. We give an incomplete majority graph, say M' , and a voting tree, say T' , such that the set of the candidates returned by Algorithm *StrongWin* [6] applied to T' and M' is empty, while there is a candidate that is a necessary winner for T' and M' .

Assume we have 6 candidates, say A, B, C, D, E , and F , and that the incomplete majority graph M' has the same edges as the incomplete majority graph M considered in the proof of Theorem 11 plus the following edges: $A >_m F, F >_m B$, and $F >_m C$. Assume that the voting tree T' is

the voting tree where the winner of the voting tree T considered in the proof of Theorem 11 plays against F . Algorithm *StrongWin* applied to M' and T' returns the empty set, since Algorithm *Win* returns a set (i.e., $\{A, F\}$) which has more than one element. However, we have shown in the proof of Theorem 11 that, for every completion of M , A does not win in T . Thus, by construction of M' , for every completion of M' , A does not win in T , and so, since $F >_m B$ and $F >_m C$ in M' , for every completion of M' , F wins in T . Therefore, F is the necessary winner for M' and T' . Hence, there is a necessary winner for M' and T' but Algorithm *StrongWin* says there is none. \square

However, even if algorithm *Win* may be incorrect, the set of winners returned by this algorithm may be useful in the search for the possible/necessary winners. In fact, this is a superset of the set of the possible winners for M and T . Moreover, if Algorithm *StrongWin* applied to M and T returns a candidate, this is indeed a necessary winner for M and T .

THEOREM 13. *Let M be an incomplete majority graph, T a voting tree, and W the set of the candidates returned by Algorithm *Win* [6]. Then $W \supseteq \text{Poss}(M, T)$.*

PROOF. Let A be a candidate. We want to show that, if $A \in \text{Poss}(M, T)$, then $A \in W$. Assume that $A \notin W$. Then, by definition of Algorithm *Win*, for every subtree T' of T where $A \in \text{root}(\text{left}(T'))$ (resp., $A \in \text{root}(\text{right}(T'))$), we have $A <_m C$, for every $C \in \text{root}(\text{right}(T'))$ (resp., for every $C \in \text{root}(\text{left}(T'))$). This holds for every completion of M . Therefore, for every completion of M , A is a loser in T and thus $A \notin \text{Poss}(M, T)$. \square

COROLLARY 13.1. *Let M be an incomplete majority graph, T a voting tree, and W the set of the candidates returned by Algorithm *Win* [6]. If $|W| = 1$, then $W = \text{Nec}(M, T)$.*

PROOF. If $W = \{A\}$, then, by Theorem 13 and by the fact that $\text{Poss}(M, T)$ cannot be empty, we have that $\text{Poss}(M, T) = \{A\}$ and thus $\text{Nec}(M, T) = \{A\}$. \square

5. CONCLUSIONS

In the setting of voting trees with missing preferences and/or uncertain agenda, we have closed all open questions about the relation between the notions of winners for the profile (which are the winners we ultimately want to compute) and the corresponding notions for the majority graph (which are polynomial to find). More precisely, the overall results are summarized in Table 2, where we have inserted our new results in those of Table 1, which described the previous state of the art.

	SVT	VT
Possible Schwartz winners	\neq [8]	\neq
Necessary Schwartz winners	\neq	\neq
Possible Condorcet	$=$ [8]	$=$
Necessary Condorcet	$=$ [8]	$=$
Possible winners	\neq	\neq
Necessary winners	\neq	\neq

Table 2: New state of the art about winners computed from the majority graph or profile.

Where we see an equality in the table, it means that we can safely and correctly work from the majority graph. This

is both more compact and efficient. This happens for possible/necessary Condorcet winners. Instead, where we see an inequality, it means that using the majority graph may give us an upper or lower approximation of the desired set of winners. However, this approximation can be found in polynomial time. The approximation is closer to the correct notion in the case of simple voting trees, since algorithms *Win* and *StrongWin* are more correct for simple voting trees than for voting trees.

6. ACKNOWLEDGMENTS

This work has been partially supported by the MIUR PRIN 20089M932N project “Innovative and multi-disciplinary approaches for constraint and preference reasoning”. We would like to thank Jerome Lang for many stimulating discussions.

7. REFERENCES

- [1] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate. *Journal of the ACM*, 54(3), 2007.
- [2] R. Farquharson. *Theory of voting*. Yale University Press, New Haven, 1969.
- [3] M. JoséHerrero and S. Srivastava. Implementation via backward induction. *Journal of Economic Theory*, 56(1):70 – 88, 1992.
- [4] K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proceedings of IJCAI’05 Multidisciplinary Workshop on Advances in Preference Handling*, 2005.
- [5] J. Lang, M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Winner determination in sequential majority voting. In *Proceedings of IJCAI’07*, pages 1372–1377. AAAI Press, 2007.
- [6] M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Determining winners in weighted sequential majority voting: incomplete profiles w.r.t. majority graphs. In *Proceedings of CLIMA VIII*, 2007.
- [7] M. S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Incompleteness and incomparability in preference aggregation. In *Proceedings of IJCAI’07*, pages 1464–1469. AAAI Press, 2007.
- [8] 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 KR’08*, pages 571–578. AAAI Press, 2008.
- [9] A. D. Procaccia, A. Zohar, Y. Peleg, and J. S. Rosenschein. Learning voting trees. In *Proceedings of AAAI*, pages 110–115, 2007.
- [10] M. Trick. Small binary voting trees. In *Proceedings of COMSOC’06*, pages 500–511, 2006.
- [11] V. Vassilevska. Fixing a tournament. In *Proceedings of AAAI’10*, 2010.
- [12] T. Walsh. Complexity of terminating preference elicitation. In *Proceedings of AAMAS’08*, pages 967–974, 2008.
- [13] L. Xia and V. Conitzer. Determining possible and necessary winners under common voting rules given partial orders. In *Proceedings of AAAI’08*, pages 196–201. AAAI Press, 2008.

Tight Bounds for Strategyproof Classification

Reshef Meir
reshef.meir@mail.huji.ac.il

Shaul Almagor
shaul.almagor@gmail.com

Assaf Michaely
assafmichaely@gmail.com

Jeffrey S. Rosenschein
jeff@cs.huji.ac.il
School of Computer Science and Engineering
The Hebrew University of Jerusalem

ABSTRACT

Strategyproof (SP) classification considers situations in which a decision-maker must classify a set of input points with binary labels, minimizing expected error. Labels of input points are reported by self-interested agents, who may lie so as to obtain a classifier more closely matching their own labels. These lies would create a bias in the data, and thus motivate the design of *truthful* mechanisms that discourage false reporting.

We here answer questions left open by previous research on strategyproof classification [12, 13, 14], in particular regarding the best approximation ratio (in terms of social welfare) that an SP mechanism can guarantee for n agents. Our primary result is a lower bound of $3 - \frac{2}{n}$ on the approximation ratio of SP mechanisms under the shared inputs assumption; this shows that the previously known upper bound (for uniform weights) is tight. The proof relies on a result from Social Choice theory, showing that any SP mechanism must select a dictator at random, according to some fixed distribution. We then show how different randomizations can improve the best known mechanism when agents are weighted, matching the lower bound with a tight upper bound. These results contribute both to a better understanding of the limits of SP classification, as well as to the development of similar tools in other, related domains such as SP facility location.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—Multiagent Systems

General Terms

Theory, Algorithms, Economics

Keywords

Mechanism design, Classification, Game theory

1. INTRODUCTION

Approximate mechanism design without money (AMDw/oM) is a rapidly growing area of research in game theory and multiagent systems, whose goal is the design of mechanisms for multiagent optimization problems (without the mechanisms' use of payments).

Cite as: Tight Bounds for Strategyproof Classification, Reshef Meir, Shaul Almagor, Assaf Michaely and Jeffrey S. Rosenschein, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 319–326.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

While the underlying problems (e.g., finding the median, or finding the optimal classifier) typically have efficient algorithms, these algorithms may fail in the presence of strategic behavior. Therefore we seek mechanisms that have additional game-theoretic properties (usually strategyproofness) at the expense of a suboptimal, i.e., approximate, behavior.

One particularly interesting AMDw/oM problem is the design of truthful learning algorithms, which incentivize experts to reveal their true opinions, even in cases where they disagree with one another. Within this framework, we focus on binary classification—that is, there is a set of (known) data points that our mechanism needs to classify as positive/negative. Data points can represent, for example, medical records of tumors that an expert-system has to classify as either *malignant* or *benign*. Following the standard classification literature, the classifier is selected from a predefined set of classifiers (e.g., linear separators in some space) known as the *concept class*.

Our mechanism outputs a classifier based on labels collected from n distinct experts. The goal of the mechanism is to maximize social welfare, by selecting a classifier that is close *on average* to the opinions of all experts. However, experts may disagree as to the correct label of a specific point. Furthermore, they may behave strategically, i.e., report false labels if this will bias the resulting classifier to be closer to their opinion. We are therefore interested in *strategyproof* (SP) classification mechanisms, where no agent (expert) can “gain” by lying. As a result, the outcome is just an *approximation* of the optimal classifier, i.e., the selected classifier makes more errors than the optimal one. We seek the best possible approximation ratio that can be guaranteed using SP mechanisms.

1.1 Motivation

Note that the restriction to a predefined concept class is an important part of the problem. Without it, we could simply classify each data point separately. However, as rigorously demonstrated in the machine learning literature, it is precisely this restriction that enables us to generalize, i.e., to apply the outcome classifier on new, unseen, cases. Previous papers on SP classification and learning (see the next section) cover real-world examples where the need to generalize justifies this restriction.

Nevertheless, SP classification might be required also for one-time *decision making*. The following is an example showing how concept class restrictions can be derived from external constraints.

An example.

Consider a situation in which two or more parties (the agents of our scenario) are in a conflict regarding the ownership of a certain piece of land. The property is abundant with resources in various

locations (the data points), and the parties may attribute different (possibly negative) importance to each resource. A neutral arbitrator agrees to hear them out and divide the field between them in a way that will maximize the average utility of all the involved parties. It is reasonable to assume that this division has some constraints, for example, that the border has to be a straight line, or that it has to pass through a specific location. This leaves us with a (large, possibly infinite) set of borders, or classifiers, from which the arbitrator may choose. Knowing how their reported preferences affect the decision, each party may misreport its true evaluation of each resource, in an attempt to achieve a favorable outcome.

1.2 Related Work

Strategyproof classification.

The first paper on SP classification was by Meir, Procaccia, and Rosenschein [12], who studied a highly restricted case in which only two classifiers are available. The authors proposed a simple deterministic 3-approximation mechanism, and proved that no better (deterministic) SP mechanisms exist. They further demonstrated a randomized SP mechanism that guarantees an approximation ratio of 2, and that this bound is also tight.

We follow an extension of this model outlined by the same authors in [13], where arbitrary concept classes can be used, but the same set of data points is still *shared* by all agents. Notably, no bounded approximation ratio can be guaranteed by deterministic SP mechanisms, but the authors show how selecting a random agent as a dictator guarantees an approximation ratio of 3, and one that is even better ($3 - \frac{2}{n}$) when agents are non-weighted. However, it is unknown whether better randomized mechanisms exist.

A similar model without the shared inputs assumption has also been studied, showing mainly negative results [14]. Using results from social choice theory, the authors showed that deterministic SP mechanisms cannot guarantee any useful approximation ratio. They further conjectured that a similar reduction can be used to supply a lower bound for randomized mechanisms, but failed to supply one that does not require further technical assumptions.

Approximate mechanism design without money.

Mechanisms that deal with strategic behavior of agents have been proposed recently for a large range of applications. While certain restrictions may allow the design of optimal SP mechanisms [19], often this is not the case, and approximation is a must. Outside the classification domain, SP learning algorithms were studied for both clustering [17] and regression [16, 4]. Other mechanisms have been proposed for facility location (see e.g., [1, 11], and [18], which also provides a clear overview of the field), matching [2, 6], resource allocation [8, 9] and more. As our motivating example shows, problems in one domain can sometimes be formalized in other domains as well. There are also interesting similarities between some of the results and techniques in those various domains.

Other related work.

A closely related, yet different, challenge is *adversarial classification* [10, 3, 5]. Here the underlying assumption is that labels are chosen intentionally to hamper the mechanism (for example to avoid spam detection), whereas in our setting the agents are rational, rather than adversarial. Another difference is that the goal of SP classification is to preclude untruthful behavior in the first place, and not to cope with it.

1.3 Our Contribution

We close the gap left open by [13], matching their $3 - \frac{2}{n}$ upper

bound for the non-weighted case with an equal lower bound, thus proving its tightness. The proof relies on the fact that every SP mechanism must be (randomly) dictatorial on a subdomain, thereby showing that the technical assumptions in [14] can be eliminated.

We then consider the weighted case, giving three different SP mechanisms for two agents that beat the known upper bound of 3. While the approximation ratio of the first mechanism is still suboptimal ($\sqrt{5}$), it is based on simple heuristics, and shows an interesting relation to the golden ratio. The other two mechanisms guarantee 2-approximation, thereby matching both the upper and lower bounds for two non-weighted agents. Finally, we present a new mechanism for any set of weighted agents, with a guaranteed approximation ratio of $3 - \frac{2}{n}$, thereby improving the previously known upper bound and matching it with the lower bound.

Omitted proofs are available in the full version of this paper.¹

2. MODEL AND NOTATIONS

2.1 Classification

We adopt the shared input model presented in [13], being consistent where possible with their notations. We refer the reader to previous work on SP classification [12, 13, 14] for more details.

We typically denote sets and their elements by $A = \{a_1, a_2, \dots\}$, and vectors by $\mathbf{a} = (a(1), a(2), \dots)$. $\Delta(A)$ contains all probability distribution vectors over the set A . $\llbracket E \rrbracket$ denotes the indicator variable of the expression E . To facilitate reading, subscripts are sometimes omitted when clear from the context.

Classifiers.

A *classification setting* is a pair $\langle \mathcal{X}, \mathcal{C} \rangle$, where \mathcal{X} (the input space) is some finite set, and \mathcal{C} (the concept class) contains functions of the form $c : \mathcal{X} \rightarrow \{-, +\}$. In the land-ownership problem for example, \mathcal{C} contains all the allowed partitions of the territory.

An *instance* of the setting $\langle \mathcal{X}, \mathcal{C} \rangle$ is a tuple defined as $S = \langle X, I, \{Y_i\}_{i \in I}, \mathbf{w} \rangle$, where $X \in \mathcal{X}^k$ is the (public) set of data points to be classified, I is the set of $n \geq 2$ agents, $Y_i : X \rightarrow \{-, +\}$ is the “correct” labeling according to agent i , and $w_i \in \mathbb{R}$ is her weight ($\sum_{i \in I} w_i = 1$). Y_i is referred to as agent i ’s *type*, and it is private information. We denote the partial dataset of agent i by $S_i = \langle X, Y_i \rangle$. S contains all possible datasets over the input space \mathcal{X} . Let $\mathcal{S}_{n,k}$ be the set of all possible datasets S such that $|I| = n$, $|X| = k$. We also allow the limit case $k = \infty$, in which case $Y_i : \mathcal{X} \rightarrow [0, 1]_{\mathbb{Q}}$ states the (rational) positive fraction on each input point. \mathcal{S} contains all datasets (finite and infinite).

The *private risk* of a classifier $c \in \mathcal{C}$ is defined as the fraction of agent i ’s dataset that is misclassified by c , i.e.,

$$\mathbf{R}_i(c, S) = \frac{1}{k} \sum_{\langle x, y \rangle \in S_i} \llbracket c(x) \neq y \rrbracket = \frac{1}{k} \sum_{x \in X} \llbracket c(x) \neq Y_i(x) \rrbracket.$$

As $\mathbf{R}_i(c, S)$ can be seen as a measure of *dissatisfaction* that i suffers due to outcome c , the *global risk* $\mathbf{R}_I(c, S)$ measures the social welfare, i.e. the (dis)satisfaction of the entire society. It is defined as a weighted average over all agents,

$$\mathbf{R}_I(c, S) = \sum_{i \in I} w_i \cdot \mathbf{R}_i(c, S) = \frac{1}{k} \sum_{i \in I} \sum_{x \in X} w_i \llbracket c(x) \neq Y_i(x) \rrbracket.$$

Let $\mathbf{p} \in \Delta(\mathcal{C})$ be a lottery over the concept class \mathcal{C} , that assigns the probability $p(\mathbf{w})$ to the concept $c_{\mathbf{w}}$. For simplicity we treat \mathbf{p} as if it is a classifier, and extend the risk to lotteries linearly, i.e., $\mathbf{R}(\mathbf{p}, S) = \sum_{\mathbf{w} \in \mathcal{X}} p(\mathbf{w}) \cdot \mathbf{R}(c_{\mathbf{w}}, S)$.

¹ftp://ftp.cs.huji.ac.il/users/jeff/aamas11meir.pdf

We denote by $ERM(S) \in \mathcal{C}$ (for Empirical Risk Minimizer) the concept that makes the smallest number of errors on S . c_i is a shorthand for $ERM(S_i)$ when S is clear from the context.

Mechanisms.

A *randomized mechanism* is a function $\mathcal{M} : \mathcal{S} \rightarrow \Delta(\mathcal{C})$, i.e., that for every input dataset of any size, outputs a lottery over classifiers. We denote by $\mathcal{M}(S)$ or $\mathbf{p}_{\mathcal{M}(S)}$ (or just \mathbf{p} when \mathcal{M}, S are clear from the context) the outcome of the randomized mechanism \mathcal{M} on the input dataset S .

Note that we can define a mechanism using a lottery \mathbf{d} over several other mechanisms $\mathcal{M}_1, \mathcal{M}_2, \dots$, where $\mathbf{p}_{\mathcal{M}(S)}(c)$ equals $\sum d(j) \mathbf{p}_{\mathcal{M}_j(S)}(c)$. We define the following properties:

A *dictator* mechanism is identified with a single agent i . For any S , \mathcal{M} returns $c_i(S)$ with probability 1.

A *duple* is a mechanism that assigns probability 0 to all concepts, except (at most) two.

A *random-dictator* (RD) mechanism is identified with a lottery $\mathbf{d} \in \Delta(I)$ over dictator mechanisms. This distribution may depend on agent weights, if relevant. The two following RD mechanisms are notable special cases:

- The *weighted random dictator* (WRD) mechanism returns $c_i(S)$ w.p. w_i .
- The *heaviest dictator* (HD) mechanism always returns $c_h(S)$, where $h = \operatorname{argmax}_{i \in I} w_i$. Ties are broken in favor of the agent with the higher index, thus h is uniquely defined.

A *random-dictator-duple* (RDD) mechanism is a lottery over dictators and duples.

A mechanism is said to be an *L-approximation* mechanism if its expected risk is at most L times the optimal risk. Formally, for every dataset S

$$\mathbf{R}_I(\mathcal{M}(S), S) \leq L \cdot \mathbf{R}_I(c^*(S), S).$$

A mechanism is said to be *strategyproof* (SP), if no agent can gain (in expectation) by lying. Formally, for every dataset S , agent i , and alternative labels $\bar{S}_i = \langle X, \bar{Y}_i \rangle$,

$$\mathbf{R}_i(\mathcal{M}(S), S) \leq \mathbf{R}_i(\mathcal{M}(S_{-i}, \bar{S}_i), S).$$

Note that duples and dictator mechanisms are always SP. Moreover, RDs and RDDs are also SP.²

Intuitively, good mechanisms are both SP and have a low approximation ratio; thus, we are interested in the best possible approximation ratio that can be achieved by randomized SP mechanisms. The following bounds are known:

THEOREM 1 (MEIR, PROCACCIA AND ROSENSCHEIN [12]). *If $|\mathcal{C}| = 2$, then there is a randomized SP mechanism that guarantees a 2-approximation ratio. Furthermore, no SP mechanism can do better.*

Thus for classes of two functions, SP mechanisms are thoroughly understood. For general concept classes, there are upper bounds:

THEOREM 2 (MEIR, PROCACCIA AND ROSENSCHEIN [13]). *For any concept class \mathcal{C} , the WRD mechanism guarantees a 3-approximation ratio. If all agents have equal weight, then the approximation ratio is $3 - \frac{2}{n}$.*

²This is since duples and dictators are SP in *dominant strategies*, not just in expectation, and therefore any combination of them (as long as it does not depend on labels) is still SP.

There are examples showing that these are the best approximation ratios that WRD can guarantee. However, it has been unknown whether there are *other* SP mechanisms that are better. Our work comes to answer this question. We make use of two additional properties of classification mechanisms.

Let $a \cdot S$ be a *duplication* of S , i.e., every data point in S appears exactly a times in $a \cdot S$, with the same labels. A mechanism is *consistent* if for all $a \in \mathbb{N}$, $S \in \mathcal{S}$, $\mathcal{M}(S) = \mathcal{M}(a \cdot S)$.

A probability distribution \mathbf{p} is *μ -granular* if all probabilities $p(c)$ are multiples of μ , i.e., if there is some integer vector \mathbf{q} such that $\mathbf{q} \cdot \mu = \mathbf{p}$. A mechanism is said to be *μ -granular* if for all S , $\mathcal{M}(S)$ is μ -granular. Note that when we deal with mechanisms that are implemented on digital computers, it is useful to assume that they will be μ -granular for some μ .

2.2 Voting

Our proofs make extensive use of voting functions and their relations with classification mechanisms. We bring here the definitions relevant to our needs. For a more detailed background on voting, see e.g., [15].

In a voting scenario there is a set of voters (agents) I , and a finite set of candidates \mathcal{C} . Each voter has a strict preference order R_i over all candidates. We denote by $c \succ_i c'$ the fact that voter i prefers c over c' . A *preference profile* $R = (R_1, \dots, R_n)$ contains the preference order of each voter (agent). Let \mathcal{R}^n be the set of all possible preference profiles for n voters, $\mathcal{R} = \bigcup_{n \geq 2} \mathcal{R}^n$.

A *randomized voting rule* is a function $f : \mathcal{R} \rightarrow \Delta(\mathcal{C})$. Note that preferences are private, thus the voting rule must use the orders reported by the agents. The definitions of a duple, RD and RDD also apply to voting rules. While the definition of manipulation in deterministic voting rules is straightforward (i.e., there is an agent that can gain by reporting false preferences), it does not apply as-is to randomized rules. This is since the preferences of agent i over lotteries of candidates are not uniquely defined by R_i . To that end, we must introduce cardinal (dis)utilities.³

A utility scale $u_i \in \mathbb{R}^{|\mathcal{C}|}$ fits order R_i if for all $c, c' \in \mathcal{C}$,

$$u_i(c) < u_i(c') \iff c \succ_i c'.$$

We adopt the same notation to classification settings, meaning that the risk of c is higher than the risk of c' .

A *manipulation* in f (by Gibbard) consists of a profile R , a utility scale u_i that fits R_i , and an alternative order R'_i , such that i gains according to u_i (formally, that $u_i(f(R)) > u_i(f(R_{-i}, R'_i))$). A voting rule is *strategyproof* (SP) if there are no manipulations in f .

THEOREM 3 (GIBBARD [7]). *Let f be a randomized voting rule. If f is SP, then it is a lottery over duples and dictatorial rules.*

3. RESULTS

3.1 Multiple Agents with Uniform Weights

In this section we match the upper bound of $3 - \frac{2}{n}$ with a lower bound, thus proving it is tight.

We use a simple input space with three input points $\mathcal{X} = \{x, y, z\}$. There are 3 classifiers, $\mathcal{C} = \{c_x, c_y, c_z\}$, where $c_w(w') = "+"$ for $w = w'$ and $"-"$ otherwise. When both the agent and the dataset are clear from the context, we use the shorthand $r(w) = \mathbf{R}_i(c_w, S)$.

THEOREM 4. *Let \mathcal{M} be an SP mechanism for the scenario $(\mathcal{X}, \mathcal{C})$. Then for any $\bar{\epsilon} > 0$ and any $|I| = n \geq 2$, there is an*

³For consistency with the risk, we treat lower utility as *better*.

instance S with uniform weights such that

$$\mathbf{R}_I(\mathcal{M}(S), S) > \left(3 - \frac{2}{n} - \bar{\epsilon}\right) \mathbf{R}_I(c^*(S), S).$$

Also, if \mathcal{M} is either consistent or μ -granular, then we can find such a dataset which is finite, and has $k = O\left(\frac{1}{\bar{\epsilon}}, \frac{1}{\mu}\right)$ data points.

We will restrict the allowed datasets as follows. First, X contains exactly k data points on each input point, i.e., $3k$ data points in total. We denote by $\bar{k}_i(w)$, $\underline{k}_i(w)$ the number of positive and negative labels for each point. We further restrict the labels of each agent, such that: one input point of \mathcal{X} is all negative (i.e., $\bar{k}_i(\cdot) = 0$); one is all positive (i.e., $\underline{k}_i(\cdot) = k$); and the third has at least one label of each (i.e., $1 \leq \bar{k}_i(\cdot) \leq k - 1$).

We refer to this third point as the *contingent point*.⁴ Clearly, \mathcal{M} is still SP w.r.t. the restricted case.

The risk of each classifier can be simply written (e.g., for c_x) as

$$r(x) = \mathbf{R}_i(c_x, S) = \frac{1}{3k} (\underline{k}_i(x) + \bar{k}_i(y) + \bar{k}_i(z)).$$

Note that every partial dataset S_i is now identified with a strict preference order R_i over \mathcal{C} (for ease of exposition, assume $R_i = (c_x \succ_i c_y \succ_i c_z)$), and a rational number $\alpha_i \in (0, 1)$ which is the fraction of negative labels on the contingent point y .

To see this, observe that

$$r(x) = \frac{1 - \alpha_i}{3}; r(y) = \frac{1 + \alpha_i}{3}; r(z) = \frac{3 - \alpha_i}{3}. \quad (1)$$

Consequently, c_x, c_z classify the contingent point (which is y in this case) as negative, and c_y classifies it as positive.

We can therefore write each S_i as $\langle R_i, \alpha_i \rangle$.

Our proof sketch can be summarized as follows:

1. Give a simpler, normalized presentation of the risk scale.
2. Show that \mathcal{M} is monotonic.
3. Show that any (monotonic) SP mechanism must ignore the value of α .
4. Thus \mathcal{M} is actually a randomized voting rule over \mathcal{C} .
5. Since \mathcal{M} is SP, it is an RDD.
6. Duples are bad, so \mathcal{M} is almost entirely an RD.
7. We show a dataset S on which RD mechanisms have a close to $3 - \frac{2}{n}$ approximation ratio.

Crucially, all steps except the last one (Lemma 11) are independent of agent weights.

Proof of Theorem 4. The preference order of agent i over lotteries in a given setting S , is completely defined by her risk scale, i.e., by the vector $\mathbf{r} = (r(x), r(y), r(z))$. Note that the risk of lottery \mathbf{p} according to risk scale \mathbf{r} is the inner product $\mathbf{R}_i(\mathbf{p}, S) = \mathbf{r} \cdot \mathbf{p}$.

DEFINITION 1. Two risk scales \mathbf{r}, \mathbf{t} are equivalent, if for any two outcomes $\mathbf{p}, \mathbf{p}' \in \Delta(\mathcal{C})$,

$$\mathbf{r} \cdot \mathbf{p} < \mathbf{r} \cdot \mathbf{p}' \iff \mathbf{t} \cdot \mathbf{p} < \mathbf{t} \cdot \mathbf{p}',$$

i.e., if they induce the same order over outcomes.

⁴For infinite datasets with $k = \infty$ this means that the contingent point must have a non-zero fraction of each sign.

LEMMA 5 (NORMALIZATION). Let $S_i = \langle R_i, \alpha_i \rangle$, then the risk scales $\mathbf{r} = (r(x), r(y), r(z))$ and $\mathbf{t} = (0, \alpha_i, 1)$ are equivalent.

Proof. We denote by $\delta(w) = p(w) - p'(w)$. Note that

$$\delta(x) + \delta(y) + \delta(z) = 0. \quad (2)$$

In addition, it holds from (1) that

$$\frac{r(y) - r(x)}{r(z) - r(x)} = \frac{1 + \alpha_i - (1 - \alpha_i)}{3 - \alpha_i - (1 - \alpha_i)} = \frac{2\alpha_i}{2} = \alpha_i. \quad (3)$$

$$\begin{aligned} \mathbf{p} \cdot \mathbf{r} < \mathbf{p}' \cdot \mathbf{r} & \iff \\ 0 > p(x)r(x) + p(y)r(y) + p(z)r(z) & \\ - (p'(x)r(x) + p'(y)r(y) + p'(z)r(z)) & \\ = \delta(x)r(x) + \delta(y)r(y) + \delta(z)r(z) & \\ = \delta(x)r(x) + \delta(y)r(y) + \delta(z)r(z) & \\ - (\delta(x) + \delta(y) + \delta(z))r(x) & \quad \text{(from (2))} \\ = \delta(y)(r(y) - r(x)) + \delta(z)(r(z) - r(x)) & \iff \\ 0 > \delta(y) \frac{r(y) - r(x)}{r(z) - r(x)} + \delta(z) & \quad \text{(division by a positive number)} \\ = \delta(y)\alpha_i + \delta(z) & \quad \text{(from (3))} \\ = \delta(x)t(x) + \delta(y)t(y) + \delta(z)t(z) = \mathbf{p} \cdot \mathbf{t} - \mathbf{p}' \cdot \mathbf{t}, & \end{aligned}$$

thus $\mathbf{p} \cdot \mathbf{t} < \mathbf{p}' \cdot \mathbf{t}$, as required. \square

Due to Lemma 5, we can work with the normalized risk scale \mathbf{t} instead of \mathbf{r} . This also holds for utility scales of voting functions.

REMARK 1. Normalization only works for a fixed scale \mathbf{r} . If \mathbf{t} is the normalized scale of \mathbf{r} , it is not true for example that $\mathbf{p} \cdot \mathbf{t} > \mathbf{p}' \cdot \mathbf{t}$ derives $\mathbf{p} \cdot \mathbf{r} > \mathbf{p}' \cdot \mathbf{r}$.

The following notations are used in our next two lemmas. Let $S_i = \langle R_i, \alpha \rangle$, $S'_i = \langle R_i, \alpha' \rangle$. Assume w.l.o.g. that $R_i = (x \succ_i y \succ_i z)$ (i.e., x has the lowest risk for i). Let $\mathbf{p} = \mathcal{M}(S)$ and $\mathbf{p}' = \mathcal{M}(S')$ denote the outcome of the mechanism on both datasets. Let \mathbf{t} and $\delta(w)$ as in Lemma 5.

Since \mathcal{M} is SP, we have the following constraints:

1. $\mathbf{R}_i(\mathbf{p}, S) \leq \mathbf{R}_i(\mathbf{p}', S)$ (otherwise, i can easily gain by reporting S'_i instead of S_i).
2. $\mathbf{R}_i(\mathbf{p}, S') \geq \mathbf{R}_i(\mathbf{p}', S')$ (otherwise, i can gain by reporting S_i instead of S'_i).

We use $r(w)$ and $r'(w)$ as shorthand for $\mathbf{R}_i(w, S)$ and $\mathbf{R}_i(w, S')$, respectively.

The next lemma shows that SP mechanisms must be “monotone”, i.e., adding more positive labels to a point can only increase the probability that it will be classified as positive.

LEMMA 6 (MONOTONICITY). If $\alpha < \alpha'$, then $p(y) \geq p'(y)$.

Proof. From the first constraint we have that $\mathbf{p} \cdot \mathbf{r} \leq \mathbf{p}' \cdot \mathbf{r}$. From Lemma 5 we can replace \mathbf{r} with the normalized risk \mathbf{t} , and thus

$$\begin{aligned} \mathbf{p} \cdot \mathbf{t} & \leq \mathbf{p}' \cdot \mathbf{t} & \iff \\ p(y)\alpha + p(z) & \leq p'(y)\alpha + p'(z) & \iff \\ \delta(y)\alpha & \leq -\delta(z) & \quad (4) \end{aligned}$$

Similarly, from the second constraint we have that

$$\delta(y)\alpha' \geq -\delta(z) \quad (5)$$

Taking the two inequalities together,

$$\begin{aligned}
\delta(y)\alpha &\leq -\delta(z) \leq \delta(y)\alpha' && \Rightarrow \\
\alpha\delta(y) &\leq \alpha'\delta(y) && \Rightarrow \\
\delta(y) &\leq \frac{\alpha'}{\alpha}\delta(y) && \Rightarrow \quad (\text{since } \frac{\alpha'}{\alpha} > 1) \\
\delta(y) \geq 0 &\Rightarrow p(y) \geq p'(y) && \square
\end{aligned}$$

OBSERVATION 7. *If there is a manipulation under utility scale $(0, \alpha, 1)$, the same manipulation must work either for any $1 > t > \alpha$, or for any $0 < t < \alpha$. This follows directly from (4), since the inequality must hold as we change α in one of the directions.*

Our next lemma shows that the size of the positive fraction on the contingent point is irrelevant, as long as the preference order R_i is kept.

LEMMA 8 (INVARIANCE OF LABELS).

$$\mathcal{M}(S_{-i}, S_i) = \mathcal{M}(S_{-i}, S'_i).$$

Proof. We need to show that the constraints induced by strategyproofness become inconsistent unless the outcomes \mathbf{p} and \mathbf{p}' coincide. Unfortunately, the constraints that follow from α and α' will not suffice, and it is in fact possible to find a pair of outcomes that hold them. The crux lies in adding a *third* point β between the first two, showing that new constraints reach a contradiction.

We rename α' to γ , so that we have $\alpha < \beta < \gamma$. We denote the outcome of \mathcal{M} on each dataset as \mathbf{p}_α , \mathbf{p}_β , and \mathbf{p}_γ , where $\mathbf{p}_\alpha = \mathcal{M}(S_{-i}, \langle R_i, \alpha \rangle)$, etc. Rewriting (4) and reversing p_α, p_γ ,

$$(p_\gamma(y) - p_\alpha(y))\alpha \geq p_\alpha(z) - p_\gamma(z) \quad (6)$$

Using β , we similarly derive the constraints:

$$(p_\beta(y) - p_\alpha(y))\beta \leq p_\alpha(z) - p_\beta(z) \quad (7)$$

(otherwise reporting $\langle R_i, \alpha \rangle$ is a manipulation in β), and

$$(p_\gamma(y) - p_\beta(y))\gamma \leq p_\beta(z) - p_\gamma(z) \quad (8)$$

(otherwise reporting $\langle R_i, \beta \rangle$ is a manipulation in γ).

Now, assume (towards a contradiction) that $p_\alpha(y) \neq p_\gamma(y)$. From monotonicity we have that $p_\alpha(y) > p_\gamma(y)$, and strict inequality also holds for at least one of the subintervals, i.e., either $p_\alpha(y) > p_\beta(y)$ or $p_\beta(y) > p_\gamma(y)$.

$$\begin{aligned}
(p_\gamma(y) - p_\alpha(y))\alpha &\geq p_\alpha(z) - p_\gamma(z) && \text{(from (6))} \\
&= (p_\alpha(z) - p_\beta(z)) + (p_\beta(z) - p_\gamma(z)) \\
&\geq (p_\beta(y) - p_\alpha(y))\beta + (p_\gamma(y) - p_\beta(y))\gamma && \text{(from (7),(8))} \\
&> (p_\beta(y) - p_\alpha(y))\alpha + (p_\gamma(y) - p_\beta(y))\alpha \\
&\hspace{10em} \text{(from monotonicity and } \alpha < \beta, \gamma) \\
&= (p_\beta(y) - p_\alpha(y) + p_\gamma(y) - p_\beta(y))\alpha \\
&= (p_\gamma(y) - p_\alpha(y))\alpha, \text{ which is a contradiction.}
\end{aligned}$$

Thus $p_\alpha(y) = p_\gamma(y)$, i.e., $\delta(y) = 0$. From (4) and (5) it follows that $\delta(z) = 0$. Finally, from (2) we have that $\delta(x) = 0$ as well, and therefore $\mathcal{M}(S_{-i}, S_i) = \mathbf{p} = \mathbf{p}' = \mathcal{M}(S_{-i}, S'_i)$.

A subtle issue lies in the finite k case, since the proof works only for pairs α, γ that differ by at least 2 points (so there is β between them). However, for $k \geq 5$, take any $\alpha < \alpha' < \gamma < \gamma'$. We then have that $\mathbf{p}_\gamma = \mathbf{p}_\alpha = \mathbf{p}'_\gamma = \mathbf{p}'_\alpha$, i.e., the same distribution must be used at every point. \square

LEMMA 9 (REDUCTION). *\mathcal{M} is an RDD.*

Proof. This lemma completes the argument that \mathcal{M} is effectively a voting rule, and therefore subject to the known limitations of SP voting rules. It must use our assumptions on \mathcal{M} in order to bound the sample size; however, we first prove the lemma *without* these assumptions, for the limit case of $k = \infty$.

We define a voting rule f as follows. For any profile R , construct the corresponding dataset S by setting $S_i = \langle R_i, \alpha_i \rangle$ for some arbitrary $\alpha_i \in (0, 1)$. The (randomized) outcome of f is defined to be $\mathcal{M}(S)$. From Lemma 8, the choice of α_i does not affect the outcome of f .

Assume (towards a contradiction) that there is a collection of datasets \hat{S} on which \mathcal{M} is not an RDD. Let $\hat{\mathcal{R}}$ be the corresponding preference profiles to \hat{S} ; thus f is not an RDD on these profiles. From Theorem 3, f is not SP, and thus has a manipulation.

W.l.o.g., there is a manipulation (in f) for voter i , such that $x \succ_i y \succ_i z$. By scaling u_i , we can further assume that $u_i(x) = 0$, $u_i(y) = \beta$, $u_i(z) = 1$.⁵

From Observation 7 we can assume that the same manipulation works with $\beta = \frac{1}{k'}$ for some $k' \in \mathbb{N}$ (or $\beta = 1 - \frac{1}{k'}$, which is the symmetric case).

It is easy to see that if $S_i = \langle R_i, \beta \rangle$, then reporting the false labeling $S'_i = \langle R'_i, \alpha_i \rangle$ is a manipulation for agent i in \mathcal{M} :

$$\begin{aligned}
u_i(f(R)) &> u_i(f(R_{-i}, R'_i)) \Rightarrow \\
\mathbf{R}_i(\mathcal{M}(S), S) &> \mathbf{R}_i(\mathcal{M}(S_{-i}, S'_i), S),
\end{aligned}$$

since u_i is also the normalized risk scale for S_i . This is in contradiction to \mathcal{M} being SP; therefore, \mathcal{M} is an RDD.

Since $\frac{1}{\beta}$ is not bounded, we allow $\frac{k_i(y)}{k}$ to take arbitrarily small values, which is the limit case $S_{k=\infty}$.

Bounding k under the consistency assumption.

We next show how the lemma still holds for *any* k , provided that \mathcal{M} is consistent. It holds from the previous paragraph that \mathcal{M} behaves as an RDD for all datasets of size k' or more. Let $k'' \geq k$ such that $k'' = a \cdot k$ for some integer a . Now consider all a duplications of datasets of size k , i.e., all duplicated datasets $a \cdot S$ s.t. $S \in \mathcal{S}_k$. Since \mathcal{M} is an RDD for $\mathcal{S}_{k''}$, it is in particular an RDD for the duplicated datasets $a \cdot \mathcal{S}_k \subseteq \mathcal{S}_{k''}$, and from consistency also for \mathcal{S}_k .

Bounding k under the μ -granularity assumption.

We show that under this assumption, \mathcal{M} is RDD for all datasets of size $k' \geq \frac{2}{\mu}$. Denote by \mathbf{p}, \mathbf{p}' the output of \mathcal{M} on the sets S_i and S'_i , respectively, and let $\delta = \mathbf{p} - \mathbf{p}'$. Recall that the normalized utility scale of i is $(0, \beta, 1)$. Since R' is a manipulation, we have that

$$u_i(f(R)) - u_i(f(R_{-i}, R'_i)) = \beta\delta(y) + \delta(z) > 0. \quad (9)$$

We wish to show that there exists $\beta' \in [\frac{\mu}{2}, 1 - \frac{\mu}{2}]$ such that if we take $\beta = \beta'$, then R' remains a manipulation (and then k' samples suffice).

Case 1 If $\delta(z) = 0$, then from (9) we have $\delta(y) > 0$. Thus, taking $\beta = \mu$ still ensures that R' is a manipulation, since $\mu\delta(y) + \delta(z) = \mu\delta(y) > 0$.

⁵More formally, if there is a manipulation according to u_i , then from Lemma 5 the same manipulation works with the utility scale $u' = (0, \beta, 1)$, where $\beta = \frac{u_i(z) - u_i(y)}{u_i(z) - u_i(x)}$.

	S_1			$\mathbf{R}_1(c)$	$S_j, j \neq 1$			$\mathbf{R}_j(c)$
	x	y	z		x	y	z	
$\bar{k}_i(\cdot)/k$	$1 - \epsilon$	1	0		1	ϵ	0	
err of c_x	ϵ	1	0	$1 + \epsilon$	0	ϵ	0	ϵ
err of c_y	$1 - \epsilon$	0	0	$1 - \epsilon$	1	$1 - \epsilon$	0	$2 - \epsilon$

Table 1: The first row shows the positive fraction on each point in S . The next rows describe the errors that each classifier makes on each point. $\mathbf{R}_i(c, S)$ is the sum of error fractions of c over the three points in S_i .

Case 2 If $\delta(z) > 0$, then by the assumption of μ -granularity we have that $\delta(z) \geq \mu$. Also, we have the naïve bound of $\delta(y) \geq -1$. By setting $\beta = \frac{\mu}{2}$ we get $\frac{\mu}{2}\delta(y) + \delta(z) \geq -\frac{\mu}{2} + \mu = \frac{\mu}{2} > 0$.

Case 3 If $\delta(z) < 0$ then by (9) we get $\delta(y) \geq \beta\delta(y) > -\delta(z)$. Thus, we can write $-\delta(z) = a\mu$ and $\delta(y) = b\mu$ for integers $\frac{1}{\mu} \geq b > a \geq 0$. From this we get

$$\begin{aligned} \frac{-\delta(z)}{\delta(y)} &= \frac{a\mu}{b\mu} \leq \frac{a\mu}{(a+1)\mu} = \frac{a}{a+1} = 1 - \frac{1}{a+1} \\ &\leq 1 - \frac{1}{\frac{1}{\mu} + 1} = 1 - \frac{\mu}{1 + \mu} < 1 - \frac{\mu}{2}. \end{aligned}$$

Thus, we have $(1 - \frac{\mu}{2})\delta(y) + \delta(z) > 0$. \square

We introduce a small constant $\epsilon > 0$, whose value will be determined later. For now it is sufficient to require that the number of samples k would be at least $\frac{1}{\epsilon}$, so that the contingent point can have a positive fraction of ϵ or less.

LEMMA 10. *If \mathcal{M} returns a duple with some probability greater than 3ϵ , then its approximation ratio is at least 3.*

Proof. Suppose that with probability of at least 3ϵ , \mathcal{M} returns a duple over $\{c_x, c_y\}$. We define a dataset S , in which all agents label z as positive, x as negative, and y with a positive fraction of ϵ (i.e., $\bar{k}_i(z) = k$, $\bar{k}_i(x) = 0$, and $\bar{k}_i(y) = 1$).⁶ The optimal classifier $c^*(S)$ is of course c_z , with a global risk of $r^* = \frac{1}{3k}$.

However, \mathcal{M} must return c_y (or c_x) w.p. of at least 3ϵ ; thus its risk is at least $3\epsilon \cdot \mathbf{R}_I(c_y, S) = 3\epsilon \left(\frac{1}{3} + \frac{1}{k}\right) > \epsilon \geq 3 \cdot r^*$. \square

We can therefore assume that \mathcal{M} returns a random dictator w.p. of at least $1 - 18\epsilon$ (there are 6 different duples, and each one has a probability of at most 3ϵ).

LEMMA 11. *Assume all n agents have the same weight. If \mathcal{M} returns a random dictator (i.e., some lottery \mathbf{d} over agents), then the approximation ratio of \mathcal{M} is at least $3 - \frac{2}{n} - \epsilon''$, where $\epsilon'' = 2n\epsilon + 96\epsilon > 0$.*

Proof. Let i (w.l.o.g. $i = 1$) be the agent selected with the highest probability (i.e., $d(1) \geq \frac{1}{n}$). We define the dataset S as follows: $S_1 = \langle (y \succ x \succ z), 1 - \epsilon \rangle$, and for all $j \neq 1$, $S_j = \langle (x \succ y \succ z), \epsilon \rangle$. Thus the selected concept of agent 1 is $c_1 = c_y$, and the selected concept of any other agent is $c_j = c_x$ (which is also the optimal concept). The construction of S is given in Table 1. To simplify computations, we do not divide the risk by the number of points and agents, and thus the global risk is in the range $[0, 3n]$. Thus,

⁶In the limit case replace $\frac{1}{k}$ with ϵ , as any fraction is allowed.

$$\begin{aligned} r^*(S) &= \mathbf{R}_I(c_x, S) = \mathbf{R}_1(c_x, S_1) + (n-1)\mathbf{R}_j(c_x, S_j) \quad (10) \\ &= 1 + \epsilon + (n-1)\epsilon = 1 + n\epsilon, \text{ whereas} \end{aligned}$$

$$\begin{aligned} \mathbf{R}_I(c_y, S) &= \mathbf{R}_1(c_y, S_1) + (n-1)\mathbf{R}_j(c_y, S_j) \quad (11) \\ &= 1 - \epsilon + (n-1)(2 - \epsilon) = 2n - 1 - n\epsilon. \end{aligned}$$

Our RD mechanism returns $c_1 = c_y$ w.p. of $d(1) \geq \frac{1}{n}$, and the best thing it can do is return $c^* = c_x$ w.p. of $1 - \frac{1}{n}$. The risk of the mechanism can be lower-bounded as follows:

$$\begin{aligned} \mathbf{R}_I(\mathcal{M}) &\geq \frac{1}{n}\mathbf{R}_I(c_y, S) + \frac{n-1}{n}r^* \\ &\geq \frac{1}{n}(2n - 1 - n\epsilon) + \frac{n-1}{n}(1 + n\epsilon) \quad (\text{from (10),(11)}) \\ &= 2 - \frac{1}{n} - \epsilon + 1 + n\epsilon - \frac{1}{n} - \epsilon \\ &= 3 - \frac{2}{n} + (n-2)\epsilon = 3 - \frac{2}{n} + (\epsilon'' - \epsilon'') + (n-2)\epsilon \\ &= 3 - \frac{2}{n} - \epsilon'' + (2n\epsilon + 96\epsilon) + n\epsilon - 2\epsilon \\ &> 3 - \frac{2}{n} - \epsilon'' + \left(3 - \frac{2}{n} - \epsilon''\right)n\epsilon \\ &= \left(3 - \frac{2}{n} - \epsilon''\right)(1 + n\epsilon) = \left(3 - \frac{2}{n} - \epsilon''\right)r^*. \end{aligned}$$

\square

Finally, we bound the total risk of \mathcal{M} . Due to Lemma 9, the outcome of \mathcal{M} is an RDD, i.e., a lottery over all 6 possible duples, and n possible dictators. We denote by RD the event that \mathcal{M} selected any of the dictators. Note that due to Lemma 10, either $Pr(RD) \geq 1 - 18\epsilon$, or the approximation ratio of \mathcal{M} is at least 3 (and thus we are done).

Assume therefore that $Pr(RD) \geq 1 - 18\epsilon$. From Lemma 11 we have that $\mathbf{R}_I(\mathcal{M}(S), S|RD) \geq \left(3 - \frac{2}{n} - \epsilon''\right)r^*(S)$ (for S as defined in the lemma). Denote $\epsilon' = 18\epsilon$, $\tilde{\epsilon} = \epsilon'' + 6\epsilon' = (2n + 200)\epsilon$.

$$\begin{aligned} \mathbf{R}_I(\mathcal{M}(S), S) &= Pr(RD)\mathbf{R}_I(\mathcal{M}(S), S|RD) \\ &\quad + Pr(\neg RD)\mathbf{R}_I(\mathcal{M}(S), S|\neg RD) \\ &\geq Pr(RD)\mathbf{R}_I(\mathcal{M}(S), S|RD) \\ &\geq (1 - \epsilon') \left(3 - \frac{2}{n} - \epsilon''\right)r^*(S) \quad (\text{from Lemmas 10,11}) \\ &> (1 - \epsilon') \left(3 - \frac{2}{n} - \tilde{\epsilon} + 6\epsilon' - \frac{4}{n}\epsilon' - 2\tilde{\epsilon}\epsilon'\right)r^*(S) \\ &= (1 - \epsilon') \left(3 - \frac{2}{n} - \tilde{\epsilon}\right)(1 + 2\epsilon')r^*(S) \\ &= (1 + \epsilon' - 2(\epsilon')^2) \left(3 - \frac{2}{n} - \tilde{\epsilon}\right)r^*(S) \\ &> \left(3 - \frac{2}{n} - \tilde{\epsilon}\right)r^*(S). \end{aligned}$$

This concludes our proof, as for any $\tilde{\epsilon}$, we only need to set ϵ small enough (i.e., k large enough). Specifically, $k \geq \frac{1}{\epsilon} = \frac{2n+200}{\tilde{\epsilon}}$ will suffice. \blacksquare

3.2 Two Weighted Agents

In this section, we restrict our analysis to datasets that are composed of just two partial datasets. Due to [13] we know that the WRD mechanism guarantees a 3-approximation ratio in the worst-case. Moreover, we know that for this mechanism the analysis is tight when the smaller weight approaches 0. As for a lower bound,

we know from [12] that it is at least 2. Theorem 4 does not contribute anything in this case, both because weights are non-uniform, and because $3 - \frac{2}{n}$ for $n = 2$ is still 2.

Due to Lemmas 9 and 10, we know that in this case too, any SP mechanism must be an RD (with high probability), but we still have the freedom to define the probability of selecting each of the two dictators, according to their weights.

Unless explicitly stated otherwise, we assume w.l.o.g. that $w_1 \leq \frac{1}{2} \leq w_2$, and denote $w = w_1$. We consider the HD and WRD mechanisms, as described in Section 2.1. Clearly both mechanisms are SP.

Consider Theorem 2. A slight variation of its proof reveals a more accurate bound. Let $w_{\min} = \min_{i \in I} w_i$ be the weight of the lightest agents (in the two agent case, $w_{\min} = w$).

THEOREM 12. *WRD has an approximation ratio of $3 - 2w_{\min}$, and this bound is tight.*

The following lemma will be useful in the analysis of our proposed mechanisms. The proof is omitted due to space constraints.

LEMMA 13. *Let $S = \langle X, I, \{Y_i\}_{i \in I}, \mathbf{w} \rangle$ be some instance with n agents. Suppose we remove an agent (w.l.o.g. agent 1), thereby creating an instance $S' = \langle X, I', \{Y_i\}_{i \in I'}, \mathbf{w}' = (w_2, \dots, w_n) \rangle$. Let $c' = c^*(S')$ be the optimal classifier for S' ; then*

$$\mathbf{R}_I(c', S) \leq \frac{1 + w_1}{1 - w_1} \mathbf{R}_I(c^*(S), S).$$

THEOREM 14. *HD has an approximation ratio of $\frac{1+w}{1-w}$, and this bound is tight.*

Proof. The upper bound follows immediately from Lemma 13, as c' is selected by the remaining, heavier, agent. For tightness, consider the following scenario. Let $w \leq \frac{1}{2}$. There are 2 samples: $X = \{x, y\}$. Agent 1 classifies both as “-”, and agent 2 classifies x as “+” and y as “-”. There are two classifiers, $\mathcal{C} = \{c_+, c_-\}$, that classify both samples as “+” and “-”, respectively. The optimal classifier is obviously c_- , whose risk is $1 - w$. However, the heaviest dictator is agent 2, who chooses c_+ (we assume a bias for tie-breaking). The risk of c_+ is $2w + 1 - w = 1 + w$. Thus, the approximation ratio in this case is $\frac{1+w}{1-w}$. ■

Next, we combine HD and WRD into a better SP mechanism. Let $T = \frac{3-\sqrt{5}}{2}$. We define the *threshold dictator* (TD) as follows.

- The TD mechanism behaves like WRD when $w > T$ and like HD otherwise.

COROLLARY 15. *TD has a worst-case approximation ratio of $\sqrt{5}$, and this bound is tight.*

Proof. Suppose $w \leq T$. Then from Theorem 14 the approximation ratio of TD is $\frac{1+w}{1-w} \leq \frac{1+T}{1-T} = \sqrt{5}$. Now suppose $w > T$; then from Theorem 12 the approximation ratio of TD is $3 - 2w \leq 3 - 2T = \sqrt{5}$. The lower bound is achieved for $w = T$. ■

Curiously, the optimal threshold T is such that the ratio between agents’ weights is exactly Φ , the golden ratio.

A natural question is whether *even better* SP mechanisms exist, and in particular mechanisms that match the lower bound of $3 - \frac{2}{2} = 2$. Interestingly, the answer is *yes*, and we now give two examples of such mechanisms.

- The *square-weight random dictator* (SRD) mechanism returns c_i w.p. $\frac{w_i^2}{\sum_{j \in I} w_j^2}$.

THEOREM 16. *For two agents, the SRD mechanism has a worst-case approximation ratio of 2.*

Proof. We will use the following lemma, showing a reduction to a simpler problem (proof omitted).

LEMMA 17. *Consider a setting with only two concepts that disagree on all points $\{c_-, c_+\}$, and let \mathcal{M} be an RD mechanism for two agents. If \mathcal{M} guarantees L -approximation in this restricted setting (for $L \geq 2$), then \mathcal{M} is an L -approximation mechanism.*

Due to Lemma 17, we can assume that c_1, c_2 completely disagree, and that one of them is the optimal classifier c^* . Assume w.l.o.g. that $c^* = c_1$, and denote the optimal risk by r^* .

Suppose first that $w > 1 - w$. This is the easy case, as it implies that the better classifier is selected with greater probability. Assume therefore that $w \leq 1 - w$, and consider mechanism HD. In the latter case, we have that $\mathbf{R}_I(\text{HD}(S), S) = 1 - r^*$. From Theorem 14 we have that $1 - r^* \leq \frac{1+w}{1-w} r^*$, therefore

$$\begin{aligned} \mathbf{R}_I(\text{SRD}(S), S) &= \frac{w^2 \mathbf{R}_I(c_1, S) + (1-w)^2 \mathbf{R}_I(c_2, S)}{w^2 + (1-w)^2} \\ &= \frac{w^2 r^* + (1-w)^2 (1-r^*)}{w^2 + (1-w)^2} \leq \frac{w^2 r^* + (1-w)^2 \frac{1+w}{1-w} r^*}{w^2 + (1-w)^2} \\ &= \frac{w^2 r^* + (1-w)(1+w)r^*}{w^2 + (1-w)^2} = \frac{1}{2w^2 - 2w + 1} r^* \\ &\leq \frac{1}{1/2} r^* = 2r^*, \end{aligned}$$

where the last inequality exists since $2w^2 - 2w + 1$ has a minimum in $w = \frac{1}{2}$. ■

By considering Lemma 17 together with Theorem 1, it follows directly that there is another 2-approximation mechanism, using the same randomization suggested by Meir, Procaccia and Rosenschein for the two-function setting [12]. We refer to this mechanism as MPR8.⁷

3.3 More than Two Weighted Agents

In this final section we extend our results beyond the two-agent setting, describing a worst-case optimal SP mechanism for any set of weighted agents.

We first try the threshold approach. Theorem 12 supplies us with an approximation ratio of $3 - 2w_{\min}$ for the WRD mechanism. Suppose we have some SP d_{n-1} -approximation mechanism \mathcal{M}_{n-1} for $n - 1$ agents, where $d_{n-1} < 3$. We can derive an SP mechanism \mathcal{M}_n for n agents as follows: set a threshold $T_n \in (0, 1)$. If all agents weigh more than T_n , use WRD. Otherwise, remove the lightest agent and run \mathcal{M}_n on the remaining data.

THEOREM 18. *Mechanism \mathcal{M}_n is SP, and has an approximation ratio of $\max \left\{ 3 - 2T_n, \frac{1+T_n}{1-T_n} d_{n-1} \right\}$.*

The proof follows directly from Lemma 13 and Theorem 12.

We can bound the worst-case approximation then, by setting T_n such that $3 - 2T_n = \frac{1+T_n}{1-T_n} d_{n-1}$. As a special case for $n = 2$, we get the TD mechanism with $\sqrt{5}$ approximation (Theorem 15). Also, we know that $d_2 = 2$ (from Theorem 16), and thus by setting the threshold for three agents to $T_3 \cong \frac{3}{20}$, we get a (roughly) $3 - \frac{6}{20} = 2 \frac{7}{10}$ approximation mechanism for three weighted agents. Similar threshold mechanisms can be iteratively derived for any

⁷The mechanism, applied to our scenario, would select the lighter and heavier agents w.p. of $\frac{w}{2-2w}$ and $\frac{2-3w}{2-2w}$, respectively.

number of agents. While this mechanism already beats the upper bound of 3, it does not match the lower bound of $3 - \frac{2}{n}$.

We finally turn to describing our last mechanism, which either generalizes or beats all previous mechanisms for SP classification with shared inputs. Let $p'_i = \frac{w_i}{2(1-w_i)}$, and $\alpha_w = \frac{1}{\sum_{i \in I} p'_i}$.

- The *convex-weight random dictator* (CRD) mechanism, returns c_i w.p. $p_i = \alpha_w p'_i$.

THEOREM 19. *The CRD mechanism has an approximation ratio of $\alpha_w + 1$, which is at most $3 - \frac{2}{n}$.*

We omit the proof due to space constraints. However, we note that it is based on the convexity of the weight function, giving rise to the name of the mechanism. When applied to two agents, the CRD mechanism is similar (but not identical) to the MPR8 mechanism, and can therefore be seen as a generalization of it. Moreover, all the upper bounds in [12, 13], as well as the ones in this paper, follow as special cases from Theorem 19.

4. DISCUSSION

Our results have two primary implications on strategyproof classification. On the negative side, we have shown that the use of dictators is necessary if one wants to maintain truthfulness in learning algorithms, even when randomization is allowed. This means in particular that the previously known bounds for SP classification with uniform weights are tight.

On the positive side, we show that while dictators play a key role in SP classification, non-trivial selection of the dictator can lead to improvements in the approximation ratio of the mechanism. We demonstrated how simple threshold heuristics can be used to safely discard low-weight agents, thus improving the worst-case approximation ratio (although it is still suboptimal). Our main positive result is the CRD mechanism, which matches the lower bound for SP classification and therefore cannot be further improved. In addition to generalizing all previously known upper bounds for the shared input setting (from [12, 13]), our result shows that the uniform weight case is also the most difficult, and a better approximation ratio can be achieved as weights become more biased in favor of some agents.

The learning-theoretic setting.

An important issue is the possibility to generalize from sampled data, and apply the result classifier on unseen data from the same distribution (a task known as *supervised learning*). It is shown in Section 3 in [13] how the WRD mechanism can be extended in such a way to a learning-theoretic setting. We note that all of our mechanisms can be applied directly to the learning-theoretic setting, making the same strategic assumptions described in [13].

Future research.

Perhaps more important than the specific bounds we proved, our results and techniques may aid in improving the understanding of randomized approximation mechanisms in other domains. Some mechanisms for facility location [1] are based on ideas similar to the WRD mechanism; our insights can be used to improve their weighted versions. Also, our impossibility proof tackles rather general issues, such as continuity and private information. This may also help in the study of lower bounds in other domains.

Other future directions may include the study of new types of strategic behaviors in learning problems, and providing a more formal picture of the relations between seemingly unrelated Approximated Mechanism Design (without money) problems.

Acknowledgments

This work was partially supported by Israel Science Foundation grant #898/05, and Israel Ministry of Science and Technology grant #3-6797.

5. REFERENCES

- [1] N. Alon, M. Feldman, A. D. Procaccia, and M. Tennenholtz. Strategyproof approximation of the minimax on networks. *Mathematics of Operations Research*, 35(3):513–526, 2010.
- [2] I. Ashlagi, F. Fischer, I. Kash, and A. D. Procaccia. Mix and match. In *Proc. of 11th EC*, pages 305–314, 2010.
- [3] N. Dalvi, P. Domingos, Mausam, S. Sanghai, and D. Verma. Adversarial classification. In *Proc. of 10th KDD*, pages 99–108, 2004.
- [4] O. Dekel, F. Fischer, and A. D. Procaccia. Incentive compatible regression learning. *Journal of Computer and System Sciences*, 76:759–777, 2010.
- [5] O. Dekel and O. Shamir. Good learners for evil teachers. In *Proc. of 26th ICML*, pages 216–223, 2009.
- [6] S. Dughmi and A. Ghosh. Truthful assignment without money. In *Proc. of 11th EC*, pages 325–334, 2010.
- [7] A. Gibbard. Manipulation of schemes that mix voting with chance. *Econometrica*, 45:665–681, 1977.
- [8] M. Guo and V. Conitzer. Strategy-proof allocation of multiple items between two agents without payments or priors. In *Proc. of 9th AAMAS*, pages 881–888, 2010.
- [9] M. Guo, V. Conitzer, and D. Reeves. Competitive repeated allocation without payments. In *Proc. of 5th WINE*, pages 244–255, 2009.
- [10] D. Lowd, C. Meek, and P. Domingos. Foundations of adversarial machine learning. Manuscript, 2007.
- [11] P. Lu, X. Sun, Y. Wang, and Z. A. Zhu. Asymptotically optimal strategy-proof mechanisms for two-facility games. In *Proc. of 11th EC*, pages 315–324, 2010.
- [12] R. Meir, A. D. Procaccia, and J. S. Rosenschein. Strategyproof classification under constant hypotheses: A tale of two functions. In *Proc. of 23rd AAAI*, pages 126–131, 2008.
- [13] R. Meir, A. D. Procaccia, and J. S. Rosenschein. Strategyproof classification with shared inputs. In *Proc. of 22nd IJCAI*, pages 220–225, 2009.
- [14] R. Meir, A. D. Procaccia, and J. S. Rosenschein. On the limits of dictatorial classification. In *Proc. of 9th AAMAS*, pages 609–616, 2010.
- [15] B. Peleg. Game-theoretic analysis of voting in committees. In K. Arrow, A. Sen, and K. Suzumura, editors, *Handbook of Social Choice and Welfare, Vol. 1*, chapter 8. Elsevier Science B., 2002.
- [16] J. Perote and J. Perote-Peña. Strategy-proof estimators for simple regression. *Mathematical Social Sciences*, 47:153–176, 2004.
- [17] J. Perote-Peña and J. Perote. The impossibility of strategy-proof clustering. *Economics Bulletin*, 4(23):1–9, 2003.
- [18] A. D. Procaccia and M. Tennenholtz. Approximate mechanism design without money. In *Proc. of 10th EC*, pages 177–186, 2009.
- [19] J. Schummer and R. V. Vohra. Mechanism design without money. In N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, editors, *Algorithmic Game Theory*, chapter 10. Cambridge University Press, 2007.

A Double Oracle Algorithm for Zero-Sum Security Games on Graphs

Manish Jain*, Dmytro Korzhyk[†], Ondřej Vaněk⁺, Vincent Conitzer[†],
Michal Pěchouček⁺, Milind Tambe*

* Computer Science Department, University of Southern California, Los Angeles, CA. 90089
{manish.jain,tambe}@usc.edu

[†] Department of Computer Science, Duke University, Durham, NC. 27708
{dima,conitzer}@cs.duke.edu

⁺ Department of Cybernetics, Czech Technical University, Prague. Czech Republic.
{vanek,pechoucek}@agents.felk.cvut.cz

ABSTRACT

In response to the Mumbai attacks of 2008, the Mumbai police have started to schedule a limited number of inspection checkpoints on the road network throughout the city. Algorithms for similar security-related scheduling problems have been proposed in recent literature, but security scheduling in networked domains when targets have varying importance remains an open problem at large. In this paper, we cast the network security problem as an attacker-defender zero-sum game. The strategy spaces for both players are exponentially large, so this requires the development of novel, scalable techniques.

We first show that existing algorithms for approximate solutions can be arbitrarily bad in general settings. We present RUGGED (*Randomization in Urban Graphs by Generating strategies for Enemy and Defender*), the first scalable optimal solution technique for such network security games. Our technique is based on a double oracle approach and thus does not require the enumeration of the entire strategy space for either of the players. It scales up to realistic problem sizes, as is shown by our evaluation of maps of southern Mumbai obtained from GIS data.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence

General Terms

Algorithms, Security, Performance

Keywords

Game theory, Double oracle, Zero-sum games, Minimax equilibrium

1. INTRODUCTION

Securing urban city networks, transportation networks, computer networks and other critical infrastructure is a large and growing

Cite as: A Double Oracle Algorithm for Zero-Sum Security Games on Graphs, Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, Milind Tambe, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 327–334.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

area of concern. The key challenge faced in these domains is to effectively schedule a *limited number of resources* to protect against an intelligent and adaptive attacker. For example, a police force has limited personnel to patrol, operate checkpoints, or conduct searches. The adversarial aspect poses significant challenges for the resource allocation problem. An intelligent attacker may observe the strategy of the defender, and then plan more effective attacks. Predictable scheduling of defender resources can be exploited by the attacker. *Randomization* has thus been used to keep attackers at bay by increasing the uncertainty they face.

Game theory offers a principled way of achieving effective randomization. It models the varying preferences of both the defender and the attacker, and allows us to solve for optimal strategies. Recent work has also used and deployed game-theoretic techniques in real-world attacker-defender scenarios, for example, ARMOR [13] and IRIS [10].

In this paper, we model an urban network security problem as a game with two players: the defender and the attacker. The pure strategies of the defender correspond to allocations of resources to edges in the network—for example, an allocation of police checkpoints to roads in the city. The pure strategies of the attacker correspond to paths from any *source* node to any *target* node—for example, a path from a landing spot on the coast to the airport.

The strategy space of the defender grows exponentially with the number of available resources, whereas the strategy space of the attacker grows exponentially with the size of the network. For example, in a fully connected graph with 20 nodes and 190 edges, the number of defender actions for only 5 resources is $\binom{190}{5} \approx 2$ billion, while the number of possible attacker paths without any cycles is $\approx 6.6 \times 10^{18}$. Real-world networks are significantly larger, e.g., a simplified graph representing the road network in southern Mumbai has more than 250 nodes (intersections) and 600 edges (streets), and the security forces can deploy tens of resources.

We model the scenario as a zero-sum game, where the attacker gets a positive payoff in case of a successful attack and 0 otherwise, and the payoff to the defender is the negative of the attacker's payoff. Our goal is to find a minimax strategy for the defender, that is, a strategy that minimizes the maximum expected utility that the attacker can obtain.¹ The extremely large size of the games

¹Because in this work, we assume the game to be zero-sum, a minimax strategy is equivalent to a Stackelberg strategy (where the defender finds the optimal mixed strategy to commit to); moreover, via von Neumann's minimax theorem [12] (or linear programming duality), minimax strategies also correspond exactly to Nash equi-

inhibits the direct application of standard methods for finding minimax strategies. Thus, we propose a double-oracle based approach that does not require the *ex-ante* enumeration of all pure strategies for either of the players. We propose algorithms for both the defender’s and the attacker’s oracle problems, which are used iteratively to provide pure-strategy best responses for both players. While we present NP-hardness proofs for the oracle problems for both players, the entire approach remains scalable in practice, as is shown in our experiments.

We also provide experimental results on real-city networks, specifically on graphs obtained from the GIS data of southern Mumbai. The graph representation of southern Mumbai has 250 nodes and 600 edges. The placement of sources and targets in the experiment was inspired by the Mumbai 2008 attacks where the targets were important economic and political centers and the sources were placed along the coast line. Our experimental results show that this problem remains extremely difficult to solve. While we show the previous approximation method to not be ready for deployment, our own techniques will need to be enhanced further for real deployments in the city of Mumbai. We believe the problem remains within reach, and is clearly an exciting and important area for continued research.

2. RELATED WORK

Game theory has been applied to a wide range of problems where one player — the evader — tries to minimize the probability of detection by and/or encounter with the other player — the patroller; the patroller wants to thwart the evader’s plans by detecting and/or capturing him. The formalization of this problem led to a family of games, often called *pursuit-evasion games* [1]. As there are many potential applications of this general idea, more specialized game types have been introduced, e.g., *hider-seeker games* [7, 9] and *infiltration games* [2] with mobile patrollers and mobile evaders; *search games* [8] with mobile patrollers and immobile evaders; and *ambush games* [14] with the mobility capabilities reversed. In the game model proposed in this paper, the evader is mobile whereas the patroller is not, just like in ambush games. However, in contrast with ambush games, we consider *targets* (termed *destinations* in ambush games) of varying importance.

Our game model is most similar to that of *interdiction games* [17], where the evading player — the attacker — moves on an arbitrary graph from one of the origins to one of the destinations (aka. *targets*); and the interdicting player — the defender — inspects one or more edges in the graph in order to detect the attacker and prevent him from reaching the target. As opposed to interdiction games, we do not consider the detection probability on edges, but we allow different values to be assigned to the targets, which is crucial for real-world applications.

Recent work has also considered scheduling multiple-defender resources using cooperative game-theory, as in *path disruption games* [3], where the attacker tries to reach a single known target. In contrast with the static asset protection problem [6], we attribute different importance to individual targets and unlike its dynamic variant [6], we consider only static target positions. Recent work in security games and robotic patrolling [4, 10] has focused on concrete applications. However, they have not considered the scale-up for both defender and attacker strategies. For example, in ASPEN, the attacker’s pure strategy space is polynomially large, since the attacker is not following any path and just chooses exactly one target to attack. Our game model was introduced by Tsai et al. [15];

however, their approximate solution technique can be suboptimal. We discuss the shortcomings of their approach in Section 4, and provide an optimal solution algorithm for the general case.

Techniques used by RUGGED are based on a double oracle approach, as proposed by McMahan et al. [11] (corresponding exactly to the notion of constraint and column generation in linear programming). This technique is intended to solve large-scale games, and is especially useful in settings where efficient algorithms for the best-response oracle problems are available. Double oracle algorithms have subsequently been applied to various pursuit-evasion games [9, 16]. While the best-response oracle problems are NP-hard in our setting (as we show in Sections 5.3 and 5.4), we give algorithms for these problems that allow the approach to still scale to realistic instances.

3. PROBLEM DESCRIPTION

A network security domain, as introduced by Tsai et al. [15], is modeled using a graph $G = (N, E)$. The attacker starts at one of the source nodes $s \in S \subset N$ and travels along a path of his choosing to any one of the targets $t \in T \subset N$. The attacker’s pure strategies are thus all the possible $s-t$ paths from any source $s \in S$ to any target $t \in T$. The defender tries to catch the attacker before he reaches any of the targets by placing k available (homogeneous) resources on edges in the graph. The defender’s pure strategies are thus all the possible allocations of k resources to edges, so there are $\binom{E}{k}$ in total. Assuming the defender plays allocation $X_i \subseteq E$, and the attacker chooses path $A_j \subseteq E$, the attacker succeeds if and only if $X_i \cap A_j = \emptyset$. Additionally, a payoff $\mathcal{T}(t)$ is associated with each target t , such that the attacker gets $\mathcal{T}(t)$ for a successful attack on t and 0 otherwise. The defender receives $-\mathcal{T}(t)$ in case of a successful attack on t and 0 otherwise. The network security domain is modeled as a complete-information zero-sum game, where the set S of sources, T of targets, the payoffs \mathcal{T} for all the targets and the number of defender resources k are known to both the players *a-priori*. The objective is to find the mixed strategy \mathbf{x} of the defender, corresponding to a Nash equilibrium (equivalently, a minimax strategy) of this *network security game*. The notation used in the paper is described in Table 1.

$G(N, E)$	Urban network graph
\mathcal{T}	Target payoff
k	Defender resources
\mathbf{X}	Set of defender allocations, $\mathbf{X} = \{X_1, X_2, \dots, X_n\}$
X_i	i^{th} defender allocation. $X_i = \{X_{ie}\} \forall e, X_{ie} \in \{0, 1\}$
\mathbf{A}	Set of attacker paths, $\mathbf{A} = \{A_1, A_2, \dots, A_m\}$
A_j	j^{th} attacker path. $A_j = \{A_{je}\} \forall e, A_{je} \in \{0, 1\}$
\mathbf{x}	Defender’s mixed strategy over \mathbf{X}
\mathbf{a}	Adversary’s mixed strategy over \mathbf{A}
$U_d(\mathbf{x}, A_j)$	Defender’s expected utility playing \mathbf{x} against A_j
Λ	Defender’s pure strategy best response
Γ	Attacker’s pure strategy best response

Table 1: Notation

4. RANGER COUNTEREXAMPLE

RANGER was introduced by Tsai et al. [15] and was designed to obtain approximate solutions for the defender for the network security game. Its main component is a polynomial-sized linear program that, rather than solving for a distribution over allocations, solves for the *marginal* probability with which the defender covers

each edge. It does this by approximating the capture probability as the sum of the marginals along the attacker’s path. It further presents some sampling techniques to obtain a distribution over defender allocations from these marginals. What was known before was that the RANGER solution (regardless of the sampling method used) is suboptimal in general, because it is not always possible to find a distribution over allocations such that the capture probability is indeed the sum of marginals on the path. In this paper, we show that RANGER’s error can be arbitrarily large.

Let us consider the example graph shown in Figure 1. This multi-graph² has a single source node, s , and two targets, t_1 and t_2 ; the defender has 2 resources. Furthermore, the payoffs \mathcal{T} of the targets are defined to be 1 and 2 for targets t_1 and t_2 respectively.

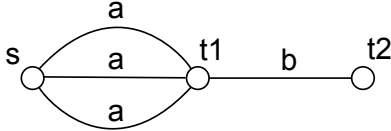


Figure 1: This example is solved *incorrectly* by RANGER. The variables a, b are the coverage probabilities on the corresponding edges.

RANGER solution:

Suppose RANGER puts marginal coverage probability a on each of the three edges between s and t_1 ,³ and probability b on the edge between t_1 and t_2 , as shown in Figure 1. RANGER estimates that the attacker gets caught with probability a when attacking target t_1 and probability $a + b$ when attacking target t_2 . RANGER will attempt to make the attacker indifferent between the two targets to obtain the minimax equilibrium. Thus, RANGER’s output is $a = 3/5, b = 1/5$, obtained from the following system of equations:

$$1(1 - a) = 2(1 - (a + b)) \quad (1)$$

$$3a + b = 2 \quad (2)$$

However, there can be no allocation of 2 resources to the edges such that the probability of the attacker being caught on his way to t_1 is $3/5$ and the probability of the attacker being caught on his way to t_2 is $4/5$. (The reason is that in this example, the event of there being a defensive resource on the second edge in the path cannot be disjoint from the event of there being one on the first edge.) In fact, for this RANGER solution, the attacker cannot be caught with a probability of more than $3/5$ when attacking target t_2 , and so the defender utility cannot be greater than $-2(1 - 3/5) = -4/5$.

Optimal solution:

Figure 2 shows the six possible allocations of the defender’s two resources to the four edges. Three of them block some pair of edges between s and t_1 . Suppose that each of these three allocations is played by the defender with probability x .⁴ Each of the other three allocations blocks one edge between s and t_1 as well as the edge between t_1 and t_2 . Suppose the defender chooses these allocations with probability y each (refer Figure 2). The probability of the attacker being caught on his way to t_1 is $\frac{2}{3}3x + \frac{1}{3}3y$, or $2x +$

²We use a multi-graph for simplicity. This counterexample can easily be converted into a similar counterexample that has no more than one edge between any pair of nodes in the graph.

³We can assume without loss of solution quality that symmetric edges will have equal coverage.

⁴Again, this can be assumed without loss of generality for symmetric edges.

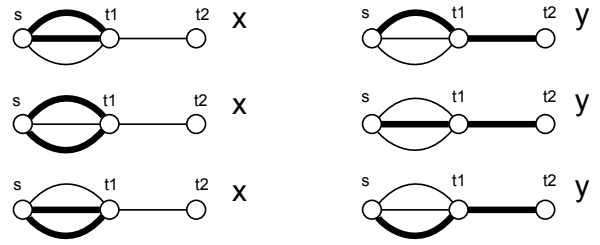


Figure 2: The possible allocations of two resources to the four edges. The blocked edges are shown in bold. The probabilities (x or y) are shown next to each allocation.

y . Similarly, the probability of the attacker being caught on his way to t_2 is $2x + 3y$. Thus, a minimax strategy for this problem is the solution of Equations (3) and (4), which make the attacker indifferent between targets t_1 and t_2 .

$$1(1 - 2x - y) = 2(1 - 2x - 3y) \quad (3)$$

$$3x + 3y = 1 \quad (4)$$

The solution to the above system is $x = 2/9, y = 1/9$, so that the expected attacker utility is $4/9$. Thus, the expected defender utility is $-4/9$, which is higher than the expected defender utility of at most $-4/5$ resulting from using RANGER.

RANGER sub-optimality:

Suppose the payoff $\mathcal{T}(t_2)$ of target t_2 in the example above was $H, H > 1$. The RANGER solution in this case, again obtained using Equations 1 and 2, would be $a = \frac{(H+1)}{(2H+1)}, b = \frac{(H-1)}{(2H+1)}$.

Then, consider an attacker who attacks the target t_2 by first going through one of the three edges from s to t_1 uniformly at random (and then on to t_2). The attacker will fail to be caught on the way from t_1 to t_2 with probability $(1 - b)$, given that the defender’s strategy is consistent with the output of RANGER. Even conditional on this failure, the attacker will fail to be caught on the way from s to t_1 with probability at least $1/3$, because the defender has only 2 resources. Thus, the probability of a successful attack on t_2 is at least $(1 - b)(1/3)$, and the attacker’s best-response utility is at least:

$$\frac{H(1 - b)}{3} = \frac{H(H + 2)}{3(2H + 1)} > \frac{H(H + 0.5)}{3(2H + 1)} = \frac{H}{6} \quad (5)$$

Thus, the true defender utility for any strategy consistent with RANGER is at most $-\frac{H}{6}$.

Now, consider another defender strategy in which the defender always blocks the edge from t_1 to t_2 , and also blocks one of the three edges between s and t_1 uniformly at random. For such a defender strategy, the attacker can reach t_1 with probability $2/3$, but cannot reach target t_2 at all. Thus, the attacker’s best-response utility in this case is $2/3$. Therefore, the optimal defender utility is at least $-2/3$. Therefore, any solution consistent with RANGER is at least $\frac{H/6}{2/3} = \frac{H}{4}$ suboptimal. Since H is arbitrary, RANGER solutions can be arbitrarily suboptimal. This motivates our exact, double-oracle algorithm, RUGGED.

5. DOUBLE-ORACLE APPROACH

In this section, we present RUGGED, a double-oracle based algorithm for network security games. We also analyze the computational complexity of determining best responses for both the defender and the attacker, and, to complete the RUGGED algorithm, we give algorithms for computing the best responses.

5.1 Algorithm

The algorithm RUGGED is presented as Algorithm 1. \mathbf{X} is the set of defender allocations generated so far, while \mathbf{A} is the set of attacker paths generated so far. $\text{CoreLP}(\mathbf{X}, \mathbf{A})$ finds an equilibrium (and hence, minimax and maximin strategies) of the two-player zero-sum game consisting of the sets of pure strategies, \mathbf{X} and \mathbf{A} , generated so far. CoreLP returns \mathbf{x} and \mathbf{a} , which are the current equilibrium mixed strategies for the defender and the attacker over \mathbf{X} and \mathbf{A} respectively. The *defender oracle* (DO) generates a defender allocation Λ that is a best response for the defender against \mathbf{a} . (This is a best response among *all* allocations, not just those in \mathbf{X} .) Similarly, the *attacker oracle* (AO) generates an attacker path Γ that is a best response for the attacker against \mathbf{x} .

Algorithm 1 Double Oracle for Urban Network Security

1. Initialize \mathbf{X} by generating arbitrary candidate defender allocations.
 2. Initialize \mathbf{A} by generating arbitrary candidate attacker paths.
 - repeat**
 3. $(\mathbf{x}, \mathbf{a}) \leftarrow \text{CoreLP}(\mathbf{X}, \mathbf{A})$.
 - 4a. $\Lambda \leftarrow DO(\mathbf{a})$.
 - 4b. $\mathbf{X} \leftarrow \mathbf{X} \cup \{\Lambda\}$.
 - 5a. $\Gamma \leftarrow AO(\mathbf{x})$.
 - 5b. $\mathbf{A} \leftarrow \mathbf{A} \cup \{\Gamma\}$.
 - until** convergence
 7. Return (\mathbf{x}, \mathbf{a})
-

The double oracle algorithm thus starts with a small set of pure strategies for each player, and then grows these sets in every iteration by applying the best-response oracles to the current solution. Execution continues until convergence is detected. Convergence is achieved when the best-response oracles of both the defender and the attacker do not generate a pure strategy that is better for that player than the player's strategy in the current solution (holding the other player's strategy fixed). In other words, convergence is obtained if, for both players, the reward given by the best-response oracle is no better than the reward for the same player given by the *CoreLP*.

The correctness of best-response-based double oracle algorithms for two-player zero-sum games has been established by McMahan et al [11]; the intuition for this correctness is as follows. Once the algorithm converges, the current solution must be an equilibrium of the game, because each player's current strategy is a best response to the other player's current strategy—this follows from the fact that the best-response oracle, which searches over all possible strategies, cannot find anything better. Furthermore, the algorithm must converge, because at worst, it will generate all pure strategies.

5.2 CoreLP

The purpose of *CoreLP* is to find an equilibrium of the restricted game consisting of defender pure strategies \mathbf{X} and attacker pure strategies \mathbf{A} . Below is the standard formulation for computing a maximin strategy for the defender in a two-player zero-sum game.

$$\max_{U_d^*, \mathbf{x}} U_d^* \quad (6)$$

$$\text{s.t. } U_d^* \leq U_d(\mathbf{x}, A_j) \quad \forall j = 1, \dots, |\mathbf{A}| \quad (7)$$

$$\mathbf{1}^T \mathbf{x} = 1 \quad (8)$$

$$\mathbf{x} \in [0, 1]^{|\mathbf{X}|} \quad (9)$$

The defender's mixed strategy \mathbf{x} , defined over \mathbf{X} , and utility U_d^* are the variables for this problem. Inequality (7) is family of constraints; there is one constraint for every attacker path A_j in \mathbf{A} . The

function $U_d(\mathbf{x}, A_j)$ is the expected utility of the attacker path A_j . Given A_j , the probability that the attacker is caught is the sum of the probabilities of the defender allocations that would catch the attacker. (We can sum these probabilities because they correspond to disjoint events.) More precisely, let z_{ij} be an indicator for whether allocation X_i intersects with path A_j , that is,

$$z_{ij} = \begin{cases} 1 & \text{if } X_i \cap A_j \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

These z_{ij} are not variables of the linear program; they are parameters that are determined at the time the best responses are generated. Then, the probability that an attacker playing path A_j is caught is $\sum_i z_{ij} x_i$, and the probability that he is *not* caught is $\sum_i (1 - z_{ij}) x_i$. Thus, the payoff function $U_d(\mathbf{x}, A_j)$ for the defender for choosing a mixed strategy \mathbf{x} when the attacker chooses path A_j is given by Equation (11), where $T(t_j)$ is the attacker's payoff for reaching t_j .

$$U_d(\mathbf{x}, A_j) = -T(t_j) \cdot \left(\sum_i (1 - z_{ij}) x_i \right) \quad (11)$$

The dual variables corresponding to Inequality (7) give the attacker's mixed strategy \mathbf{a} , defined over \mathbf{A} . The expected utility for the attacker is given by $-U_d^*$.

5.3 Defender Oracle

This section concerns the best-response oracle problem for the defender. The *Defender Oracle* problem is stated as follows: generate the defender pure strategy (resource allocation) Λ allocating k resources over the edges E that maximizes the defender's expected utility against a given attacker mixed strategy \mathbf{a} over paths \mathbf{A} .

Defender Oracle problem is NP-hard: We show this by reducing the set cover problem to it. **The Set-Cover problem:** Given are a set U , a collection \mathcal{S} of subsets of U (that is, $\mathcal{S} \subseteq 2^U$), and an integer k . The question is whether there is a cover $\mathcal{C} \subseteq \mathcal{S}$ of size k or less, that is, $\bigcup_{c \in \mathcal{C}} c = U$ and $|\mathcal{C}| \leq k$. We will use a modification of this well-known NP-hard problem so that \mathcal{S} always contains all singleton subsets of U , that is, $x \in U$ implies $\{x\} \in \mathcal{S}$. This modified problem remains NP-hard.

THEOREM 1. *The Defender Oracle problem is NP-hard, even if there is only a single source and a single target.*

PROOF. Reduction from Set-Cover to Defender Oracle: We convert an arbitrary instance of the set cover problem to an instance of the defender oracle problem by constructing a graph G with just 3 nodes, as shown in Figure 3. The graph G is a multi-graph⁵ with just three nodes, so that $\mathbf{N} = \{s, v, t\}$, where s is the only source and t is the only target (with arbitrary positive value). There are up to $|\mathcal{S}|$ loop edges adjacent to node v ; each loop edge corresponds to a unique non-singleton subset in \mathcal{S} . There are $|U|$ edges between s and v , each corresponding to a unique element in U . There are also $|U|$ edges between v and t , each corresponding to a unique element in U . The attacker's paths correspond to the elements in U . A path that corresponds to $u \in U$ starts with the edge between s and v that corresponds to u , then loops through all the edges that correspond to non-singleton subsets in \mathcal{S} that contain u , and finally ends with the edge between v and t that corresponds to u . Hence, any two paths used by the attacker can only intersect at the loop edges. The probabilities that the defender places on these paths are arbitrary positive numbers. We now show that set U can be covered with k subsets in $\mathcal{S} \subseteq 2^U$ if and only if the defender can block all of the attacker's paths with k resources in the corresponding defender oracle problem instance.

⁵Having a multi-graph is not essential to the NP-hardness reduction.

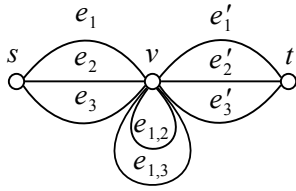


Figure 3: A defender oracle problem instance corresponding to the SET-COVER instance with $U = \{1, 2, 3\}$, $\mathcal{S} = \{\{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}\}$. Here, the attacker’s mixed strategy uses three paths: $(e_1, e_{1,2}, e_{1,3}, e'_1)$, $(e_2, e_{1,2}, e'_2)$, $(e_3, e_{1,3}, e'_3)$. Thus, the SET-COVER instance has a solution of size 2 (for example, using $\{1, 2\}$ and $\{1, 3\}$); correspondingly, with 2 resources, the defender can always capture the attacker (for example, by covering $e_{1,2}, e_{1,3}$).

The “if” direction: If the defender can block all the paths used by the attacker with k resources, then the set U can be covered with $\mathcal{C} \subseteq \mathcal{S}$, where $|\mathcal{C}| = k$ and is constructed as follows. If the defender places a resource on a loop edge, then \mathcal{C} includes the non-singleton subset in \mathcal{S} that corresponds to that loop edge. If the defender blocks any other edge then \mathcal{C} includes the corresponding singleton subset.

The “only if” direction: If there exists a cover \mathcal{C} of size k , then the defender can block all the paths by placing a defensive resource on every loop edge that corresponds to a non-singleton subset in \mathcal{C} , and placing a defensive resource on the corresponding edge out of s for every singleton subset in \mathcal{C} . \square

Formulation: The defender oracle problem, described below, can be formulated as a mixed integer linear program (MILP). The objective of the MILP is to identify the allocation that covers as many attacker paths as possible, where paths are weighted by the product of the payoff of the target attacked by the path and probability of attacker choosing it. (In this formulation, probabilities a_j are not variables; they are provided by *CoreLP*.) In the formulation, $\lambda_e = 1$ indicates that we assign a resource to edge e , and $z_j = 1$ indicates that path A_j (refer Table 1) is blocked by the allocation.

$$\max_{z, \lambda} \quad -\sum_j (1 - z_j) a_j \mathcal{T}_{t_j} \quad (12)$$

$$\text{s.t.} \quad z_j \leq \sum_e A_{j,e} \lambda_e \quad (13)$$

$$\sum_e \lambda_e \leq k \quad (14)$$

$$\lambda_e \in \{0, 1\} \quad (15)$$

$$z_j \in [0, 1] \quad (16)$$

THEOREM 2. *The MILP described above correctly computes a best-response allocation for the defender.*

PROOF. The defender receives a payoff of $-\mathcal{T}(t_j) a_j$ if the attacker successfully attacks target t_j using path A_j , and 0 in the case of an unsuccessful attack. Hence, if we make sure that $1 - z_j = 1$ if path A_j is not blocked, and 0 otherwise, then the objective function (12) correctly models the defender’s expected utility. Inequality (13) ensures this: its right-hand side will be at least 1 if there exists an edge on the path A_j that defender is covering, and 0 otherwise. z_j need not be restricted to take an integer value because the objective is increasing with z_j and if the solver can push it above 0, it will choose to push it all the way up to 1. Therefore, if we let Λ correspond to the set of edges covered by the defender, z_j will be set by the solver so that:

$$z_j = \begin{cases} 1 & \text{if } \Lambda \cap A_j \neq \emptyset \Leftrightarrow \exists e | \lambda_e = A_{j,e} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (17)$$

Inequality (14) enforces that the defender covers at most as many edges as the number of available resources k , and thus ensures feasibility. Hence, the above MILP correctly captures the best-response oracle problem for the defender. \square

PROPOSITION 1. *For any attacker mixed strategy, the defender’s expected utility from the best response provided by the defender oracle is no worse than the defender’s equilibrium utility in the full zero-sum game.*

PROOF. In any equilibrium, the attacker plays a mixed strategy that minimizes the defender’s best-response utility; therefore, if the attacker plays any other mixed strategy, the defender’s best-response utility can be no worse. \square

5.4 Attacker Oracle

This section concerns the best-response oracle problem for the attacker. The **Attacker Oracle** problem is to generate the attacker pure strategy (path) Γ from some source $s \in S$ to some target $t \in T$ that maximizes the attacker expected utility given the defender mixed strategy \mathbf{x} over defender allocations \mathbf{X} .

Attacker Oracle is NP-hard: We show that the attacker oracle problem is also NP-hard by reducing 3-SAT to it.

THEOREM 3. *The Attacker Oracle problem is NP-hard, even if there is only a single source and a single target.*

PROOF. Reduction from 3-SAT to Attacker Oracle: We convert an arbitrary instance of 3-SAT to an instance of the *attacker oracle* problem as follows. Suppose the 3-SAT instance contains n variables x_i , $i = 1, \dots, n$, and k clauses. Each clause is a disjunction of three literals, where each *literal* is either a variable or the negation of the variable. Consider the following example:

$$E = (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_4) \quad (18)$$

The formula E contains $n = 4$ variables and $k = 2$ clauses.

We construct a multi-graph G^6 with $n+k+1$ nodes, v_0, \dots, v_{n+k} so that the source node is $s = v_0$, and the target node is $t = v_{n+k}$. Every edge connects some pair of nodes with consecutive indices, so that every simple path from s to t contains exactly $n+k$ edges. Each edge corresponds to a literal in the 3-SAT expression (that is, either x_i or $\neg x_i$). There are exactly three edges that connect nodes v_{i-1} and v_i for $i = 1, \dots, k$. Those three edges correspond to the three literals in the i -th clause. There are exactly two edges that connect nodes v_{k+j-1} and v_{k+j} for $j = 1, \dots, n$. Those two edges correspond to literals x_j and $\neg x_j$. An example graph that corresponds to the expression (18) is shown in Figure 4.

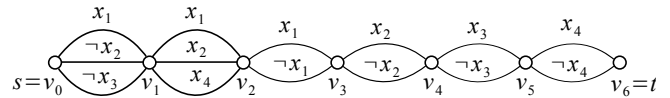


Figure 4: An example graph corresponding to the CNF formula $(x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_4)$

There are $2n$ defender pure strategies (allocations of resources), each played with equal probability of $1/(2n)$. Each defender pure strategy corresponds to a literal, and the edges that correspond to that literal are blocked in that pure strategy. In the example shown in Figure 4, the defender plays 8 pure strategies, each with probability $1/8$. Three edges are blocked in the pure strategy that corresponds to the literal x_1 (namely, the top edge between v_0 and v_1 ,

⁶We use a multi-graph for simplicity; having a multi-graph is not essential for the NP-hardness reduction.

the top edge between v_1 and v_2 , and the top edge between v_2 and v_3); only one edge is blocked in the pure strategy that corresponds to the literal $\neg x_4$ (the bottom edge between v_5 and v_6). (If it is desired that the defender always use the same number of resources, this is easily achieved by adding dummy edges.) We now show that there is an assignment of values to the variables in the 3-SAT instance so that the formula evaluates to *true* if and only if there is a path from s to t in the corresponding attacker oracle problem instance which is blocked with probability at most $1/2$.

The “if” direction: Suppose there is a path Γ from s to t that is blocked with probability at most $1/2$. Note that any path from s to t is blocked by at least one of the strategies $\{x_i, \neg x_i\}$, for all $i = 1, \dots, n$, so the probability that the path is blocked is at least $n/(2n) = 1/2$. Moreover, if for some i , the path passes through both an edge labeled x_i and one labeled $\neg x_i$, then the probability that the path is blocked is at least $(n + 1)/(2n) > 1/2$ —so this cannot be the case for Γ . Hence, we can assign the *true* value to the literals that correspond to the edges on the path Γ , and *false* to all the other literals. This must correspond to a solution to the 3-SAT instance, because each clause must contain a literal that corresponds to an edge on the path, and is thus assigned a *true* value.

The “only if” direction: Suppose there is an assignment of values to the variables such that the 3-SAT formula evaluates to *true*. Consider a simple path Γ that goes from s to t through edges that correspond to literals with *true* values in the assignment. Such a path must exist because by assumption the assignment satisfies every clause. Moreover, this path is blocked only by the defender strategies that correspond to *true* literals, of which there are exactly n . So the probability that the path is blocked is $n/(2n) = 1/2$. \square

Formulation: The attacker oracle problem can be formulated as a set of mixed integer linear programs, as described below. For every target in T , we solve for the best path to that target; then we take the best solution overall. Below is the formulation when the attacker is attacking target t_m . (In this formulation, probabilities x_i are not variables; they are values produced earlier by *CoreLP*.) In the formulation, $\gamma_e = 1$ indicates that the attacker passes through edge e , and $z_i = 1$ indicates that the allocation X_i blocks the attacker path. Equations (20) to (22) represent the flow constraints for the attacker for every node $n \in \mathbf{N}$.

$$\max_{z, \gamma} \quad \mathcal{T}_{t_m} \sum_i x_i (1 - z_i) \quad (19)$$

$$\text{s.t.} \quad \sum_{e \in \text{out}(n)} \gamma_e = \sum_{e \in \text{in}(n)} \gamma_e \quad n \neq s, t_m \quad (20)$$

$$\sum_{e \in \text{out}(s)} \gamma_e = 1 \quad (21)$$

$$\sum_{e \in \text{in}(t_m)} \gamma_e = 1 \quad (22)$$

$$z_i \geq \gamma_e + X_{ie} - 1 \quad \forall e \forall i \quad (23)$$

$$z_i \geq 0 \quad (24)$$

$$\gamma_e \in \{0, 1\} \quad (25)$$

THEOREM 4. *The MILP described above correctly computes a best-response path for the attacker.*

PROOF. The flow constraints are represented in Equations (20) to (22). The sink for the flow is the target t_m that we are currently considering for attack. To deal with the case where there is more than one possible source node, we can add a virtual source (s) to G that feeds into all the real sources. $\text{in}(n)$ represents the edges coming into n , $\text{out}(n)$ represents those going out of n . The flow constraints ensure that the chosen edges indeed constitute a path from the (virtual) source to the sink.

The attacker receives a payoff of $\mathcal{T}(t_m)$ if he attacks target t_m successfully, that is, if the path does not intersect with any defender

allocation. Hence, if we make sure that $1 - z_i = 1$ if allocation X_i does not block the path, and 0 otherwise, then the objective function (19) correctly models the attacker’s expected utility. Inequality (23) ensures this: if the allocation X_i covers some e for which $\gamma_e = 1$, then it will force z_i to be set at least to 1; otherwise, z_i only needs to be set to at least 0 (and in each case, the solver will push it all the way down to this value, which also explains why the z_i variables do not need to be restricted to take integer values). Therefore, if we let Γ correspond to the path chosen by the attacker, z_i will be set by the solver so that

$$z_i = \begin{cases} 1 & \text{if } X_i \cap \Gamma \neq \emptyset \Leftrightarrow \exists e \mid \gamma_e = X_{ie} = 1 \\ 0 & \text{otherwise} \end{cases} \quad (26)$$

It follows that the MILP objective is correct. Hence, the above MILP captures the best-response oracle problem for the attacker. \square

PROPOSITION 2. *For any defender mixed strategy, the attacker’s expected utility from the best response provided by the attacker oracle is no worse than the attacker’s equilibrium utility in the full zero-sum game.*

PROOF. In any equilibrium, the defender plays a mixed strategy that minimizes the attacker’s best-response utility; therefore, if the defender plays any other mixed strategy, the attacker’s best-response utility can be no worse. \square



Figure 5: Example graph of Southern Mumbai with 455 nodes. Sources are depicted as green arrows and targets are red bullseyes. Best viewed in color.

6. EVALUATION

In this section, we describe the results we achieved with RUGGED. We conducted experiments on graphs obtained from road network GIS data for the city of Mumbai (inspired by the 2008 Mumbai incidents [5]), as well as on artificially generated graphs. We provide two types of results: (1) Firstly, we compare the solution quality obtained from RUGGED with the solution quality obtained from RANGER. These results are shown in Section 6.1. (2) Secondly, we provide runtime results showing the performance of RUGGED when the input graphs are scaled up.⁷ The following three types of graphs were used for the experimental results:

(1) *Weakly fully connected (WFC)* graphs, denoted $G^{\text{WFC}}(N, E)$, are graphs where N is an ordered set of nodes $\{n_1, \dots, n_m\}$; $S = \{n_1\}$, $T = \{n_m\}$. For each node n_i , there exists a set of directed

⁷All experiments were run on standard desktop 2.8GHz machine with 2GB main memory.

edges, $\{(n_i, n_j) | n_i < n_j\}$, in E . These graphs were chosen because of the extreme size of the strategy spaces for both players. Additionally, there are no *bottleneck* edges, so these graphs are designed to be computationally challenging for RUGGED.

(2) *Braid*-type graphs, denoted $G^B(N, E)$, are graphs where N is a sequence of nodes n_1 to n_m such that each pair n_{i-1} and n_i is connected by 2 to 3 edges. Node n_1 is the source node. Any following node is a target node with probability 0.2, with payoff \mathcal{T} randomly chosen between 1 and 100. These graphs have a similar structure as the graph in Figure 1, and were motivated by the counterexample in Section 4.

(3) *City* graphs of different sizes were extracted from the southern part of Mumbai using the GIS data provided by OpenStreet-Maps. The placement of 2-4 targets was inspired by the Mumbai incidents from 2008 [5]; 2-4 sources were placed on the border of the graph,⁸ simulating an attacker approaching from the sea. We ran the test for graphs with the following numbers of nodes: 45, 129 and 252. Figure 5 shows a sample Mumbai graph with 252 nodes, 4 sources and 3 targets.

6.1 Comparison with RANGER

This section compares the solution quality of RUGGED and RANGER. Although we have already established that RANGER solutions can be arbitrarily bad in general, the objective of these tests is to compare the actual performance of RANGER with RUGGED. The results are given in Table 2, which shows the average and maximum error from RANGER. We evaluated RANGER on the three types of graphs — city graphs, braid graphs and weakly fully connected graphs of different sizes, fixing the number of defender resources to 2 and placing 3 targets, with varied values from the interval $[0, 1000]$. The actual defender utility from the solution provided by RANGER⁹ is computed by using the best-response oracle for the attacker with the RANGER defender strategy as input. The error of RANGER is then expressed as the difference between the defender utilities in the solutions provided by RANGER and by RUGGED.

Table 2 shows the comparison results between RANGER and RUGGED, summarized over 30 trials. It shows the percentage of trials in which RANGER gave an incorrect solution (denoted “pct”). It also shows the average and maximum error of RANGER (denoted as *avg* and *max* respectively) over these trials. It shows that while RANGER was wrong only about 1/3 of the time for Braid graphs, it gave the wrong answer in *all* the runs on the fully connected graphs. Furthermore, it was wrong 90% of the time on city graphs, with an average error of 215 units and a maximum error of 721 units. Given an average target value of 500, these are high errors indeed — indicating that RANGER is unsuitable for deployment in real-world domains.

6.2 Scale-up and analysis

This section concerns the performance of RUGGED when the input problem instances are scaled up. The experiments were conducted on graphs derived directly from portions of Mumbai’s road network. The runtime results are shown in Table 3, where the rows represent the size of the graph and the columns represent the number of defender resources that need to be scheduled. As an example of the complexity of the graph, the number of attacker paths in the Mumbai graph with 252 nodes is at least a 10^{12} , while the number of defender allocations is approximately 10^{10} for 4 resources.

⁸We placed more sources and targets into larger graphs.

⁹Because RANGER provides a solution in the form of marginal probabilities of defender allocations along edges, we used *Comb sampling* [15] to convert this into a (joint) probability distribution over defender allocations.

	City		Braid		WFC	
nodes	45	129	10	20	10	20
avg error	215	250	210	259	191	80
max error	721	489	472	599	273	117
pct	90%	100%	30%	37%	100%	100%
avg \mathcal{T}	500	500	500	500	500	500

Table 2: RANGER average and maximum error and percent of samples where RANGER provided a suboptimal solution. Target values \mathcal{T} were randomly drawn from the interval $[1, 1000]$.

	1	2	3	4
45	0.91	6.43	22.58	33.42
129	6.63	32.55	486.48	3140.23
252	17.19	626.25	2014.14	34344.70

Table 3: Runtime (in seconds) of RUGGED when the input problem instances are scaled up. These tests were done on graphs extracted from the road network of Mumbai. The rows correspond to the number of nodes in the graph whereas the columns correspond to the number of defender resources.

The game matrix for this problem cannot even be represented, let alone solved. The ability of RUGGED to compute optimal solutions in such situations, while overcoming NP-hardness of both oracles, marks a significant advance in the state of the art in deploying game-theoretic techniques.

Figure 6(a) examines the performance of RUGGED when the size of the strategy spaces for both players is increased. These tests were conducted on WFC graphs, since they are designed to have large strategy spaces. These problems have 20 to 100 nodes and up to 5 resources. The x-axis in the figure shows the number of nodes in the graph, while the y-axis shows the runtime in seconds. Different number of defender resources are represented by different curves in the graph. For example for 40 nodes, and 5 defender resources, RUGGED took 108 seconds on average.

To speed up the convergence of RUGGED, we tried to *warm-start* the algorithm with an initial defender allocation such as min-cut-based allocations, target- and source-centric allocations, RANGER allocations and combinations of these. No significant improvement of runtime was measured; in some cases, the runtime increased because of the larger strategy set for the defender.

6.3 Algorithm Dynamics Analysis

This section analyzes the anytime solution quality and the performance of each of the three components of RUGGED: the defender oracle, the attacker oracle, and the *CoreLP*. When we solve the best-response oracle problems, they provide lower and upper bounds on the optimal defender utility, as shown in Propositions 1 and 2. Figure 6(b) shows the progress of the bounds and the *CoreLP* solution for a sample problem instance scheduling 2 defender resources on a fully connected network with 50 nodes. The x-axis shows the number of iterations and the y-axis shows the expected defender utility. The graph shows that a good solution (i.e., one where the difference in the two bounds is less than ϵ) can be computed reasonably quickly, even though the algorithm takes longer to converge to the optimal solution. For example, a solution with an allowed approximation of 10 units¹⁰ can be computed in about 210 iterations, whereas 310 iterations are required to find the optimal solution. The difference between these two bounds gives an upper

¹⁰10 units is 1% of the maximum target payoff (1000).

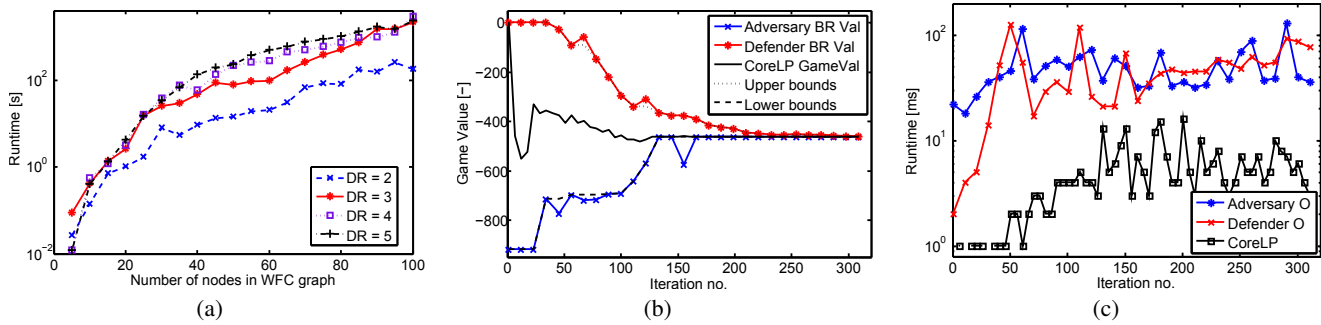


Figure 6: Results. Figure (a) shows the scale-up analysis on WFC graph of different sizes. Figure (b) shows the convergence of oracle values to the final game value and the anytime bounds. Figure (c) compares the runtimes of oracles and the core LP.

bound on the error in the current solution of the *CoreLP*; this also provides us with an *approximation* variant of RUGGED.

Figure 6(c) compares the runtime needed by the three modules in every iteration. The x-axis shows the iteration number and the y-axis shows the runtime in seconds in logarithmic scale. As expected, *CoreLP* — solving a standard linear program — needs considerably less time in each iteration than both the oracles, which solve mixed-integer programs. The figure also shows that the modules scale well as the number of iterations increases.

7. CONCLUSION AND FUTURE WORK

Optimally scheduling defender resources in a network-based environment is an important and challenging problem. Security in urban road networks, computer networks, and other transportation networks is of growing concern, requiring the development of novel scalable approaches. These domains have extremely large strategy spaces; a graph with just 20 nodes and 5 resources can have more than 2 billion strategies for both players. In this paper, we presented RUGGED, a novel double-oracle based approach for finding an optimal strategy for scheduling a limited number of defender resources in a network security environment. We showed that previous approaches can lead to arbitrarily bad solutions in such situations, and the error can be very high even in practice. We applied RUGGED to real-city maps generated from GIS data; we presented the results of applying RUGGED to the road network of Mumbai. While enhancements to RUGGED are required for deployment in some real-world domains, optimal solutions even to these problems are now within reach. The scalability of RUGGED opens up new avenues for deploying game-theoretic techniques in real-world applications.

8. ACKNOWLEDGEMENTS

This research is supported by the United States Department of Homeland Security through Center for Risk and Economic Analysis of Terrorism Events (CREATE), the Czech Ministry of Education, Youth and Sports under project number N00014-09-1-0537, the NSF CAREER grant 0953756 and IIS-0812113, ARO 56698-CI, and an Alfred P. Sloan fellowship. We thank Ron Parr, Michal Jakob and Zhengyu Yin for comments and discussions.

9. REFERENCES

- [1] M. Adler, H. Räcke, N. Sivadasan, C. Sohler, and B. Vöcking. Randomized pursuit-evasion in graphs. In *ICALP*, pages 901–912, 2002.
- [2] S. Alpern. Infiltration Games on Arbitrary Graphs. *Journal of Mathematical Analysis and Applications*, 163:286–288, 1992.
- [3] Y. Bachrach and E. Porat. Path Disruption Games. In *AAMAS*, pages 1123–1130, 2010.
- [4] N. Basilico, N. Gatti, and F. Amigoni. Leader-Follower Strategies for Robotic Patrolling in Environments with Arbitrary Topologies. In *AAMAS*, pages 500–503, 2009.
- [5] R. Chandran and G. Beitchman. Battle for Mumbai Ends, Death Toll Rises to 195. *Times of India*, 29 November 2008.
- [6] J. Dickerson, G. Simari, V. Subrahmanian, and S. Kraus. A Graph-Theoretic Approach to Protect Static and Moving Targets from Adversaries. In *AAMAS*, pages 299–306, 2010.
- [7] M. M. Flood. The Hide and Seek Game of Von Neumann. *MANAGEMENT SCIENCE*, 18(5-Part-2):107–109, 1972.
- [8] S. Gal. *Search Games*. Academic Press, New York, 1980.
- [9] E. Halvorson, V. Conitzer, and R. Parr. Multi-step Multi-sensor Hider-Seeker Games. In *IJCAI*, pages 159–166, 2009.
- [10] M. Jain, E. Kardes, C. Kiekintveld, F. Ordóñez, and M. Tambe. Security Games with Arbitrary Schedules: A Branch and Price Approach. In *AAAI*, pages 792–797, 2010.
- [11] H. B. McMahan, G. J. Gordon, and A. Blum. Planning in the Presence of Cost Functions Controlled by an Adversary. In *ICML*, pages 536–543, 2003.
- [12] J. V. Neumann. Zur Theorie der Gesellschaftsspiele. *Mathematische Annalen*, 100(1):295–320, 1928.
- [13] J. Pita, M. Jain, F. Ordóñez, C. Portway, M. Tambe, C. Western, P. Paruchuri, and S. Kraus. Using Game Theory for Los Angeles Airport Security. *AI Magazine*, 30(1), 2009.
- [14] W. Ruckle, R. Fennell, P. T. Holmes, and C. Fennemore. Ambushing Random Walks I: Finite Models. *Operations Research*, 24:314–324, 1976.
- [15] J. Tsai, Z. Yin, J. young Kwak, D. Kempe, C. Kiekintveld, and M. Tambe. Urban security: Game-theoretic resource allocation in networked physical domains. In *AAAI*, pages 881–886, 2010.
- [16] O. Vaněk, B. Bošanský, M. Jakob, and M. Pěchouček. Transiting Areas Patrolled by a Mobile Adversary. In *IEEE CIG*, pages 9–16, 2010.
- [17] A. Washburn and K. Wood. Two-person Zero-sum Games for Network Interdiction. *Operations Research*, 43(2):243–251, 1995.
- [18] Z. Yin, D. Korzhuk, C. Kiekintveld, V. Conitzer, and M. Tambe. Stackelberg vs. Nash in Security Games: Interchangeability, Equivalence, and Uniqueness. In *AAMAS*, pages 1139–1146, 2010.

Preferences and Strategies

Modeling Social Preferences in Multi-player Games

Brandon Wilson, Inon Zuckerman, and Dana Nau
Department of Computer Science,
Institute for Advanced Computer Studies, and
Institute of Systems Research
University of Maryland
College Park, MD 20742
{bswilson,inon,nau}@cs.umd.edu

ABSTRACT

Game-tree search algorithms have contributed greatly to the success of computerized players in two-player extensive-form games. In multi-player games there has been less success, partly because of the difficulty of recognizing and reasoning about the inter-player relationships that often develop and change during human game-play. Simplifying assumptions (e.g., assuming each player selfishly aims to maximize its own payoff) have not worked very well in practice.

We describe a new algorithm for multi-player games, Socially-oriented Search (SOS), that incorporates ideas from *Social Value Orientation* theory from social psychology. We provide a theoretical study of the algorithm, and a method for recognizing and reasoning about relationships as they develop and change during a game. Our empirical evaluations of SOS in the strategic board game Quoridor show it to be significantly more effective against players with dynamic interrelationships than the current state-of-the-art algorithms.

Categories and Subject Descriptors

I.2.8 [Artificial Intelligence]: Problem Solving, Control Methods, and Search—*Graph and tree search strategies*

General Terms

Economics, Algorithms

Keywords

game-tree search, multi-player games

1. INTRODUCTION

Search algorithms such as the classical Minimax search algorithm [13] are perhaps the most important component of computer programs for games of strategy. In two-person games, these algorithms have been highly successful, and there are many games in which the best computerized players perform as well or better than the best human players (e.g., [14, 15]). However, such algorithms have generally been much less successful in multi-player games—partly because they lack ways of recognizing and reasoning about

Cite as: Modeling Social Preferences in Multi-player Games, Brandon Wilson, Inon Zuckerman, and Dana Nau, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 337-344.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

inter-player relationships, which are a very influential component of the strategic behavior of human players [18]. For example, consider situations where a player has lost any “practical” chance of winning the game, but its actions might determine which of the other players will eventually win—or where a local grievance in the early stages of the game gives birth to vindictive actions in later stages. In both cases, interpersonal relationships will have considerable weight when the players reason about future courses of action.

The standard approach to dealing with this problem has been to make simplifying assumptions. For example, the Max-n algorithm [8], a generalization of Minimax to n-player games, assumes that the players will each try selfishly to maximize their own utility values, while being indifferent to the other players; and the Paranoid algorithm [16] assumes that a player’s opponents will always select the actions that are worst for that player, without considering their own chances of winning.

Such simplifying assumptions rarely hold in real-world games. For example, in games such as Risk or Diplomacy, it is very common for players to decide that one player (or a small group of players) is too strong, and join forces temporarily to fight that player (a fact that was successfully exploited by the MP-Mix algorithm presented in [20]). Such interpersonal relationships can also develop when games are played repeatedly [2].

The fundamental question is *how to describe and reason about these relationships during game play*. Our approach is to model the players’ interpersonal relationships by incorporating ideas from *Social Value Orientation* (SVO) theory into game-tree search.

Social Value Orientation (SVO) theory was first described by Messick and McClintock in [4], and [3] gives a recent overview. In this theory, the space of interpersonal behaviors is viewed as a two-dimensional spectrum in which one axis represents altruism versus aggression, and the other represents varying degrees of individualism. SVO theory states that these differences in interpersonal orientations might arise due to reasons including subjective preferences and beliefs, and ascriptive characteristics such as nationality and ethnicity (for example see Grid-Group theory [9]). These orientations often change dynamically during interactions, based on the players’ actions in the previous rounds [1].

Our new algorithm, Socially Oriented Search (SOS), uses a social-range matrix structure, based on SVO theory, to keep track of inter-player relationships and use them to guide the search. Since the interpersonal values are often unknown and change dynamically during play, we also present an on-

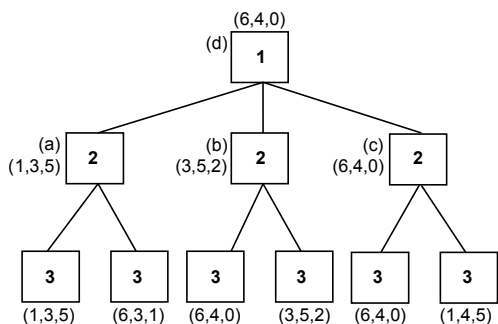


Figure 1: Propagating with the Max-n assumption

line learning technique that adapts the stored relationship values as the game progresses. In our evaluations of the algorithm in the *Quoridor* game, it played significantly better than the Max-n and Paranoid algorithms when playing against adversaries with varying interpersonal orientations.

Our main contributions can be summarized as follows:

- SOS, a novel multi-player search algorithm that considers interpersonal orientations.
- A theoretical analysis of the solution computed by SOS and conditions under which SOS converges to well-known multi-player algorithms.
- A learning version of the algorithm that learns and adapts social orientations when they are unknown.
- An empirical evaluation comparing the performance of SOS with both Max-n and Paranoid algorithms.

2. BACKGROUND AND RELATED WORK

In search algorithms for extensive-form games, when a player needs to select an action, it spans a search tree where nodes correspond to states of the game, edges correspond to moves and the root of the tree corresponds to the current state. We refer to this player as the *root player*. The leaves of the tree are evaluated according to a heuristic evaluation function and the values are propagated up to the root.

In sequential two-player games (where players alternate turns) values from the leaves are propagated according to the Minimax principle [13]. That is, in levels where it is the root player’s turn, we take the maximum among the children while in levels where it is the opponent’s turn, we take the minimum of the children.

The sequential multi-player game with n players ($n > 2$), where the players take turns in a round robin-fashion, is more complicated. The assumption is that for each node the evaluation function returns a vector H of n values where h_i estimates the merit of player i . The straightforward generalization of the two-player Minimax algorithm to the multi-player case is the Max-n algorithm [8]. It assumes that each player will try to maximize its own component of the heuristic vector, while disregarding the values of other players. Minimax can be seen as a specific instance of Max-n, where $n = 2$. An example of the Max-n propagation procedure is depicted in figure 1, where we see that each player i (player numbers are inside the nodes) selects the action which maximizes element i of the vector without considering the values of the other elements.

A different approach, called the *Paranoid* approach, was first introduced in [11] as part of a proof for search pathology in multi-player games trees, and later was presented in [16]. In this algorithm the root player takes a paranoid assumption that the opponent players will work in a coalition against him and try to minimize its heuristic value. This paranoid assumption allows the root player to reduce the game to a two-player game: the root player against a meta player that includes all the other players. Figure 2 depicts the same game tree as the previous Max-n example, but now values were propagated according to the Paranoid assumption. Here, the root player assumes that players 2 and 3 will select the action that minimizes his value.

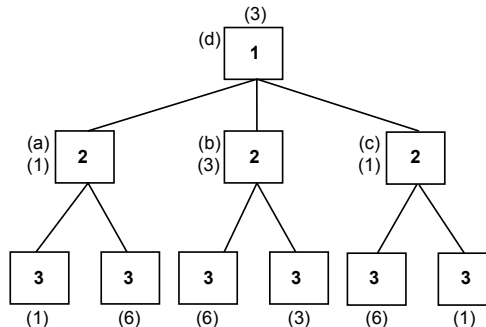


Figure 2: Propagating with Paranoid assumption

Since there is no definitive answer to which approach is better, and the answer is probably both domain and evaluation function dependent [17], a recent algorithm, *MP-Mix*, proposed switching between search strategies dynamically according to the game situation [20]. This algorithm examines the current situation and decides, according to the players’ relative strength values, whether the root player should propagate values according to the Max-n principle, the Paranoid principle, or the newly presented Directed Offensive principle. In this strategy, the root player first chooses a *target* opponent it wishes to attack. It then explicitly selects the path which results in the lowest evaluation score for the target opponent.

The above algorithms share two common assumptions: the first is that when propagating values, the adversaries use the same heuristic evaluation function as the searching player, while the second states that the adversaries are using a fixed, pre-determined preference ordering on the heuristic values. The first assumption has been dealt with through specific opponent modeling techniques. For example, the *Prob-MaxN* algorithm [19] is an extension of the Max-n algorithm where, given a set of possible evaluation functions as an input, the algorithm dynamically adapts the probabilities of each individual adversary’s membership to these prior models. In [10], when playing repeatedly against the same opponent, the authors’ algorithm learned its opponent’s weaknesses and exploited them in future games. Regarding the second assumption, these models are limited and unable to describe and reason about the more complex relationships that players might encounter in multi-player games.

3. SOCIALLY ORIENTED SEARCH

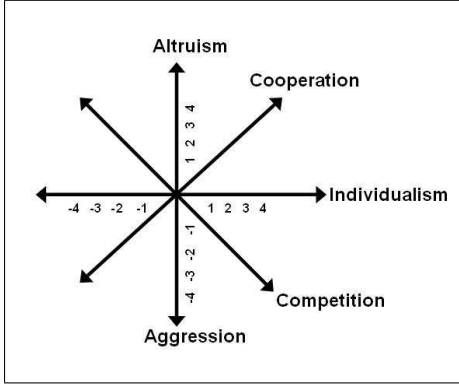


Figure 3: Social Behavior Spectrum

In this paper, we use “Social Value Orientation theory” to refer to a class of theories from the behavioral science literature [4, 3, 6] stating that people differ in their interpersonal orientations, and that they are consistent over time. Figure 3 describes a two-person preference model of the major personal and interpersonal orientations that can occur between players. In this model [6], the player’s utility is defined on the horizontal axis, and the outcome of the “other” player is on the vertical axis, and the values reflect a linear combination of payoffs to both players.

Our Socially Oriented Search algorithm utilizes a recently suggested social-range matrix model [7] that supports the description of interpersonal orientations as captured in the two-dimensional social behavior spectrum. The social matrix construct makes it possible to model “socially heterogeneous” systems where players may have different social orientations to each of the other players. The fundamental building block of our search procedure, the social-range-matrix, is defined as follows: for a game consisting of n players, the social-range matrix is an $n \times n$ matrix where element $c_{ij} \in [-1, 1]$ represents how much player i cares about player j . A value of 1 indicates a completely cooperative relationship whereas -1 indicates an aggressive one. Values in between represent varying degrees of social orientation, with 0 indicating complete apathy.¹ Given a utility vector, U , with a utility for for each player, the weighted sum of the utility vector and the i th row of the social range matrix is referred to as the *perceived utility* for player i .

3.1 The SOS Algorithm

Our Socially Oriented Search (SOS) algorithm (presented as algorithm 1) models inter-player relationships with a social-range matrix and incorporates them into the search by weighting the evaluation values in the leaf nodes to obtain the *perceived utility*. Since most games are too complex to search completely, the typical approach is to pre-select a level to cutoff the search and estimate the strength of the states at this level by a heuristic evaluation function. This evaluation is a vector where element i represents the relative strength of the game state for player i . Our algorithm transforms

¹Note that the social behaviors on the left side of the graph are deliberately excluded from our work, since these behaviors (e.g. *Masochism*, *Sado-Masochism*), are considered mental disorders.

Algorithm 1 $SOS(s, eval, d, c, p)$: Given an evaluation function, $eval$, and a social range matrix, c , compute and return the perceived utility of state s by searching to depth d . p is the player whose turn it is to move at s and N is the number of players in the game.

```
// Transform evaluation to incorporate social preferences.
// Notice the matrix-vector multiplication.
If  $d$  is 0, return  $c * eval(s)$ 
```

```
// Determine maximum of values for children nodes.
Let  $mv_1, \dots, mv_n$  be the children of  $s$  and
 $bestEvaluation = -\infty$ 
for  $i = 1, \dots, n$  do
   $v = SOS(mv_i, eval, d - 1, c, p \bmod N)$ 
  if  $v > bestEvaluation$  then
     $bestEvaluation = v$ 
     $bestMove = v$ 
  end if
end for
```

```
// Return the move with the best perceived utility
return  $bestMove$ 
```

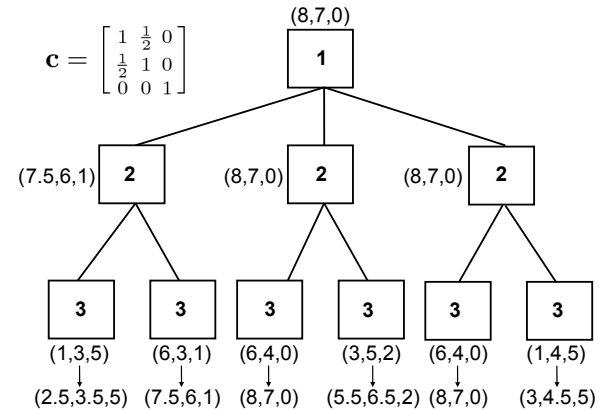


Figure 4: SOS propagation using the social-range matrix c .

this vector into the *perceived evaluation*, where element i is the dot product of the heuristic evaluation and row i of the social range matrix. The values are then propagated so that each player maximizes their element of the perceived evaluation.

Figure 4 illustrates an example of a depth-2 search that is guided by the social-range matrix c . In this particular matrix, players 1 and 2 are not completely cooperative, as they do not value each other’s utility as much as their own but they do have a positive orientation towards one another and they are ignorant of the utility for player 3. Player 3 is selfish, valuing its own utility and ignoring all others. Based on this matrix, the SOS algorithm selects the middle branch as the best decision, leading to a perceived utility of 8 for player 1.

The social range matrix guides an SOS search, selecting the move leading to the maximum perceived utility instead of selecting the move that maximizes or minimizes a single element in the evaluation vector (i.e., the Max-n and Paranoid algorithms). The ability to modify the social range

matrix makes our algorithm more flexible than either Paranoid or Max-n. In fact, one can achieve a Max-n or Paranoid search with the appropriate social range matrix. Max-n assumes that all players are selfish and ignorant of other players utility. Property 1 characterizes the features of a social-range matrix that lead to Max-n behavior.

PROPERTY 1. *Using the identity matrix with the SOS algorithm is equivalent to Max-n.*

Paranoid assumes that all players are aggressive toward the principle player with the exception of the principle player itself which is selfish. Property 2 formally characterizes the features of a social-range matrix that lead to Max-n or Paranoid behavior.

PROPERTY 2. *A matrix in which element $c_{ii} = 1$, $c_{ji} = -1, \forall j \neq i$, and all other entries are 0 represents the Paranoid assumption from the point of view of player i . Using it with the SOS algorithm is equivalent to player i using the Paranoid algorithm.*

Coalitions, where a group of players try to maximize or minimize the combined utility of another set of players, can also be represented by social range matrices, as well as an infinite number of other social preferences.

3.2 Online learning SOS

Since the social preferences of each player are not usually known ahead of time, learning algorithms can be used to estimate the social-range matrix based on previous moves. Our learning rule estimates a player’s social preferences as the average effect of the player’s last k moves. We can estimate the effect of a move from state s_1 to state s_2 as the difference in the normalized heuristic evaluations:

$$\Delta(s_1, s_2) = \frac{eval(s_2)}{max_eval} - \frac{eval(s_1)}{max_eval},$$

where max_eval is the maximum evaluation for any player’s component of the evaluation. So, given a history of k moves for player i , we estimate the i th row of the social-range matrix as the average effect of the k last moves. Higher values of k provide a richer information set to infer the social-range matrix while lower values allow the matrix to be adapted more quickly to a player whose social preferences change often over the course of the game.

4. THEORETICAL ANALYSIS

In this section we document some of the theoretical properties of the SOS algorithm. We demonstrate that SOS computes an equilibrium point, that SOS always permits immediate and shallow pruning but may also leverage deep-pruning under special circumstances, and that the behavior of SOS may converge to that of Max-n or Paranoid based on the social orientations and evaluation function.

4.1 Perceived Equilibria

For an n -player game, a player’s strategy is a plan that completely determines the player’s behavior for any potential situation that may arise during the course of the game. A strategy profile, $S = \{s_1, s_2, \dots, s_n\}$, is a set of strategies, one for each player, that completely defines all actions for the game. A strategy profile is a Nash Equilibrium if there is no player that can unilaterally change their strategy and

obtain a higher utility. The utility vector corresponding to this strategy profile is an equilibrium point. This concept assumes each player is completely selfish. For a game where players’ social orientations may vary, we have a similar concept: the *Perceived Nash Equilibrium* [7]. A strategy profile is a perceived Nash equilibrium if there is no player that can unilaterally change their strategy to obtain a higher perceived utility. Similarly, the perceived utility associated with this strategy profile is a *perceived Nash equilibrium point*.

Assuming that the social range matrix correctly represents all players’ social preferences then we can prove that our generalized search procedure identifies a *perceived equilibrium point* in a similar fashion as the proof Luckhart et al. use in showing that Max-n identifies an equilibrium [8].

THEOREM 1. *Given a perfect-information, n -player, non-cooperative game and the players’ social orientations in the form of a social-range matrix, SOS computes a perceived Nash equilibrium point.*

PROOF. Let $S = \{s_1, s_2, \dots, s_n\}$ denote the strategy profile computed by SOS that leads to the payoff vector produced by our algorithm. The payoff vector, where element i of the vector is the perceived utility for player i , propagated to the root of the search tree based on this set of strategies is $U(S)$. $U(S)$ is a perceived equilibrium point if there is no alternate strategy set $S' = s_1, s_2, \dots, s'_j, \dots, s_n$ where player j utilizes a different strategy and $U(S')[j] > U(S)[j]$. Now, assume a strategy set exists such that $U(S')[j] > U(S)[j]$, where S is the strategy set identified by our SOS. This means that there is a different set of moves that would lead to a greater perceived utility for player j . More specifically, there must be at least one node in the game tree where the move selected by j would be better if a different move were selected. This contradicts the definition of our algorithm where the move selected at each node maximizes the moving player’s perceived utility. \square

4.2 Pruning

In general, SOS is only capable of immediate and shallow pruning, however, deep-pruning is also possible under certain circumstances. Our algorithm follows the same propagation rule as Max-n with the exception of the linear transformation that is applied to the node evaluations on the search frontier. Therefore, as long as the heuristic evaluation meets the assumptions that make such pruning possible for Max-n (i.e., the minimum evaluation for each player and the maximum sum of player evaluations is known) then we can perform immediate and shallow pruning. We will not reproduce the proofs here because, with the exception of the evaluation transformation, they are identical to those for Max-n [16].

As discussed by Sturtevant and Korf [16], the Paranoid algorithm allows for deep pruning by reducing the game to two players, the principle player and a coalition of attackers cooperating against the principle player. Therefore, in addition to immediate and shallow pruning, we are also able to take advantage of alpha-beta style pruning when the social-range matrix meets the criteria of the Paranoid matrix as presented in Property 2.

Examining the Paranoid matrix, we see that reducing the game to a two-player game really means that *only* a single column of the matrix is non-zero. In other words, if column i is the non-zero column then every player’s perceived

utility is oriented around either maximizing or minimizing player i 's utility (referred to as cooperating and attacking players, respectively). These players would be max and min respectively on their turns in the corresponding two-player game tree, which in turn makes alpha-beta pruning possible [16]. This means that any situation where the social-range matrix has only one non-zero column can be deep-pruned using alpha-beta, not just when there is one cooperating player and $n - 1$ attacking players as is the case of Paranoid.

4.3 Convergence to Max-n and Paranoid

SOS does not always produce different results than Max-n or Paranoid. For example, properties 1 and 2 mention that when the social orientations match the underlying assumptions of either algorithm then the behavior will coincide with the corresponding algorithm. This section expands this concept, using features of the evaluation function as well as information about players' social orientations to determine when the social orientations can have an effect on the algorithm performance and when they can be ignored since the performance converges to that of Max-n or Paranoid.

The identity matrix is not the only matrix that will result in identical behavior to Max-n. Theorem 2 shows that scaling the diagonal of the identity matrix by any positive, real number will also produce the same behavior as Max-n.

THEOREM 2. *Given a perfect-information, n -player game and the players' social orientations in the form of a social-range matrix, c , the equilibrium strategies computed by SOS and Max-n are identical given that c is the zero matrix except for a positive, non-zero diagonal.*

PROOF. Assume that SOS does select a different strategy than Max-n. The strategies discovered by SOS and Max-n will differ if there is at least one decision in the game tree where Max-n selects a state s_1 and SOS selects state s_2 on player i 's turn. For Max-n to select s_1 over s_2 means that the i 'th component of the evaluation for s_1 must be greater than that of the evaluation for s_2 . In other words, $eval(s_1) = \{e_1, \dots, e_i, \dots, e_N\}$ and $eval(s_2) = \{e'_1, \dots, e_i - \epsilon, \dots, e'_N\}$, where ϵ is a non-zero, positive real-number such that $e_i - \epsilon$ is still a valid evaluation. Letting c_{ii} be player i 's entry in the i 'th row of the social-range matrix, SOS will choose s_2 and produce a different strategy set if:

$$(e_i - \epsilon)c_{ii} \geq (e_i)c_{ii}$$

which reduces to

$$c_{ii} \leq 0.$$

This contradicts the non-zero and positive diagonal. \square

In addition to the inter-player relationships, having knowledge of the evaluation function can also provide insights into the performance of a search algorithm. In general, an evaluation function with finer granularity (number of possible values) is better as more values means that the strength of states can be estimated more accurately. Also, Nau et al. [12] recently showed that evaluation functions with a finer granularity are less likely to exhibit pathological behavior during Minimax search. We use a granularity-based model of the evaluation function as well to analyze how the relationship between social orientations and evaluation function granularity can impact the behavior of SOS. The function we consider is of the form:

$$eval: s \rightarrow \langle e_1, e_2, \dots, e_n \rangle \mid \forall i, e_i \in \{0, \delta, 2\delta, \dots, max_eval\},$$

where max_eval is the maximum possible value for any single element of the evaluation vector, s is a state of the game, and δ represents the distance between consecutive values of the evaluations. With max_eval fixed, finer grained evaluations can be achieved by reducing δ and coarser-grained evaluations can be achieved by increasing δ .

THEOREM 3. *Given a perfect-information, n -player game, the players' social orientations in the form of a social-range matrix, c , and an evaluation function, $eval: s \rightarrow \langle e_1, e_2, \dots, e_n \rangle \mid \forall i, e_i \in \{0, \delta, 2\delta, \dots, max_eval\}$, the equilibrium strategies computed by SOS are guaranteed to be identical to those of Max-n if, for each player i :*

$$\delta > \frac{\max_{e'_j} \sum_{j \neq i} c_{ij} * e'_j - \min_{e_j} \sum_{j \neq i} c_{ij} * e_j}{c_{ii}}.$$

PROOF. Consider the turns in the game tree for an individual player. By definition, on player i 's turn, Max-n always selects the state that maximizes the i 'th component of the utility vector. This means that given that Max-n chooses s_1 over s_2 then we know their evaluations must be of the form

$eval(s_1) = \{e_1, \dots, e_i, \dots, e_N\}$ and $eval(s_2) = \{e'_1, \dots, e_i - x\delta, \dots, e'_N\}$ where x is a positive integer such that $e_i - x\delta$ is still a valid evaluation. SOS will incorporate the true social orientation of player i and make the same decision if:

$$(e_i - x\delta)c_{ii} + \sum_{j \neq i} c_{ij} * e'_j < c_{ii}e_i + \sum_{j \neq i} c_{ij} * e_j.$$

We can guarantee that this condition is true for all possible evaluation vector pairs e and e' if e' is chosen so as to maximize the summation on the left-hand side and e is chosen so as to minimize the summation on the right-hand side. This gives us:

$$(e_i - x\delta)c_{ii} + \max_{e'_j} \sum_{j \neq i} c_{ij} * e'_j < c_{ii}e_i + \min_{e_j} \sum_{j \neq i} c_{ij} * e_j.$$

Simplifying this equation we get the result displayed in Theorem 3 which must hold for all players for SOS to select the same move at every level of the search tree as Max-n. \square

Intuitively, the relationship presented in Theorem 3 states that with coarser evaluation functions, the social orientations can deviate further from the assumed relationship model of Max-n and yet the behavior of Max-n and SOS will converge. This is significant because the result that finer-grained evaluation functions make social orientations have greater effect further increases the appeal of modeling and using them with SOS.

Recognizing situations where the behavior of SOS will converge to that of Paranoid is also important. Given the social-range matrix, if we know that SOS will behave exactly as a Paranoid then the social-range matrix can be approximated by the Paranoid matrix and SOS gains the advantage of deep-pruning without sacrificing anything by ignoring the social relationships. Theorem 4 formally describes the relationship between the granularity of the evaluation function and paranoid matrix.

THEOREM 4. *Given a perfect-information, n -player game, the players' social orientations in the form of a social-range matrix, c , and an evaluation function, $eval: s \rightarrow \langle e_1, e_2, \dots, e_n \rangle \mid \forall i, e_i \in \{0, \delta, 2\delta, \dots, max_eval\}$, the equilibrium strategies computed by SOS are guaranteed to be the same as those of Paranoid if, for the principle player i :*

$$\delta > \frac{\max_{e'_j} \sum_{j \neq i}^N c_{ij} * e'_j - \min_{e_j} \sum_{j \neq i}^N c_{ij} * e_j}{c_{ii}},$$

and for every other player k :

$$\delta > \frac{\max_{e_j} \sum_{j \neq i}^N c_{kj} * e_j - \min_{e'_j} \sum_{j \neq i}^N c_{kj} * e'_j}{c_{ki}}.$$

PROOF. Similar to Theorem 3, we must show the conditions under which SOS is guaranteed to make the same choices as Paranoid during propagation. The justification for the principle player is identical to the one already shown in the proof of Theorem 3. The difference is that all other players are attacking and trying to minimize the principle player's score whereas in Max-n they are assumed to be maximizing their own utility. Therefore, given that player i is the principle player and that Paranoid chooses s_1 over s_2 on player k 's turn then we know their evaluations must be of the form

$eval(s_1) = \{e_1, \dots, e_i, \dots, e_N\}$ and $eval(s_2) = \{e'_1, \dots, e_i + x\delta, \dots, e'_N\}$ where x is a positive integer such that $e_i + x\delta$ is still a valid evaluation. SOS will incorporate the true social orientation of player i and make the same decision if:

$$(e_i + x\delta)c_{ki} + \sum_{j \neq i}^N c_{kj} * e'_j > c_{ki}e_i + \sum_{j \neq i}^N c_{kj} * e_j.$$

Now, we can guarantee that this condition is true for all possible evaluation vector pairs e and e' if e' is chosen so as to minimize the summation on the left-hand side and e is chosen so as to maximize the summation on the right-hand side. This leaves us with:

$$(e_i + x\delta)c_{ki} + \min_{e'_j} \sum_{j \neq i}^N c_{kj} * e'_j > c_{ki}e_i + \max_{e_j} \sum_{j \neq i}^N c_{kj} * e_j.$$

This reduces to the equation in Theorem 4. \square

5. EXPERIMENTAL EVALUATION

In this section we provide a discussion of the experimental analysis we performed on our algorithm. All experiments are performed on a four-player version of the game Quoridor. We show that by explicitly modeling social preferences, SOS gains a significant advantage over algorithms that make simplifying assumptions. We also show that our approach to learning the social-range matrix, although simple, is quite effective.

5.1 Game description

Quoridor² is a full-information board game for 2 or 4 players, that is played on a 9x9 grid (see figure 5). In the 4 player

²More information on that game can be found of the creator's website: <http://www.gigamic.com/>

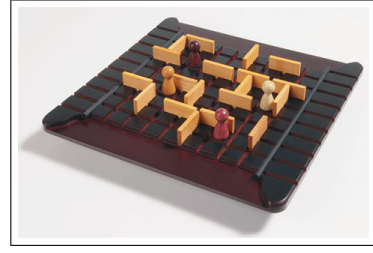


Figure 5: Quoridor board game

version, each player starts with five walls and a single pawn that is located at the middle grid location on one of the four sides of the square board. The objective is to be the first player to reach any of the grid locations on the opposite side of the board. The players move in clock-wise, sequential order, and at each turn, the player chooses to either:

1. move his pawn horizontally or vertically to one of the neighboring squares.
2. place a wall piece on the board to facilitate his progress or to impede that of his opponent.

The walls occupy the width of two grid spaces and can be used to block pathways around the board as players cannot jump over them and must navigate around them. When placing a wall, an additional rule dictates that each player has to have at least one free path to a destination on the opposing side of the board. That prevents situations in which players team-up to enclose a pawn inside 4 walls. Walls are limited and useful resource and they cannot be moved or picked up after they are placed on the board.

Quoridor is an abstract strategic game which bears some resemblance of chess and checkers. The state-space complexity of Quoridor is composed by the number of ways to place the pawns multiplied by the number of ways to place the walls, minus the number of illegal positions. Such estimation was computed in [5] for the two-player version of the game and as such places the game in the middle between Backgammon and Chess in terms of the size of the search space. Obviously that search space increases dramatically when playing the 4-player version of the game.

5.2 Experimental Design and Results

We implemented a game environment in C++. The game board was represented as a graph and Dijkstra's algorithm was used to check the legality of wall positions (i.e., to check that there exist a path to the goal). We used a simple and straightforward heuristic evaluation function that sum the total distance of each of the players to the goal. Each player seeks to minimize his own distance while maximizing the opponents' distances. Moreover, to cope with the large branching factor of the game, we limited the possible locations that a wall can be placed to a fixed radius around the pawns.

In the first set of experiments, we played the Max-n, Paranoid, and SOS algorithms against a set of random SOS players that looked only one move ahead. In the initialized stage of each game we first randomized the social orientation of the three random players, as well as the position of the search-based player (as playing order might effect the result). After the initialization stage, in order to provide a robust comparison, each player was played against the same random setting from the same position.

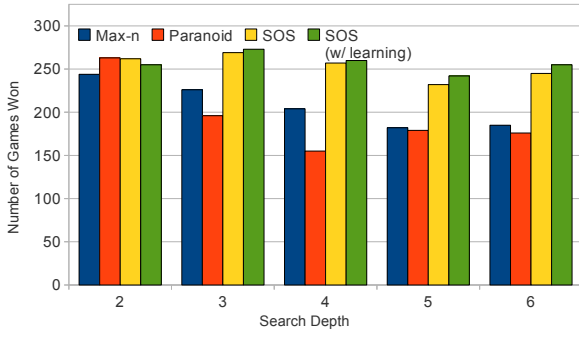


Figure 6: Results of playing Max-n, Paranoid, and SOS players in 500 Quoridor games against 3 random social-preference playing opponents. Two different SOS players were examined, one with absolute knowledge of the social-range matrix and another using naive learning method to approximate it.

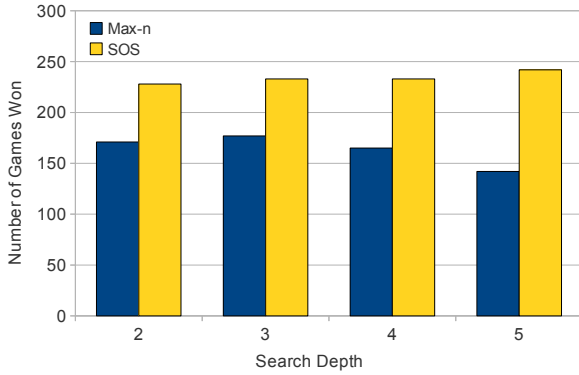


Figure 7: Results of playing Max-n and our SOS player in 500 Quoridor games against two random-preference players. The SOS player had absolute knowledge of the social-range matrix when playing.

We played both versions of the SOS algorithm, in the first we assume our Socially-oriented player had absolute knowledge of the social-range matrix in this setting. The second version was the SOS with learning ($k = 5$), where the initial social matrix values were initialized to the Max-n matrix, presented in property 1, and the algorithm adapted these values during play. Figure 6 shows that both the versions of the SOS player significantly outperforms both Max-n and Paranoid after depth 2 ($\mathcal{P} < 0.01$ in a 2-tail Z-test). Interestingly, our simple learning rule performed in a manner that is completely comparable to the SOS algorithm with the full and accurate social information.

In the second set of experiments, we replaced one of the random-preference players with a Max-n or paranoid player in order to evaluate their head-to-head performance. Thus prior to each game we randomized two random players and then plugged in an SOS player as third player. The fourth player was Max-n in the first set of experiments and Paranoid in the secondset. We ran 500 games for each depth. Figures 7 and 8 show that the non-learning algorithm, still significantly outperforms Max-n and Paranoid when competing directly against them ($\mathcal{P} < 0.01$). Although both

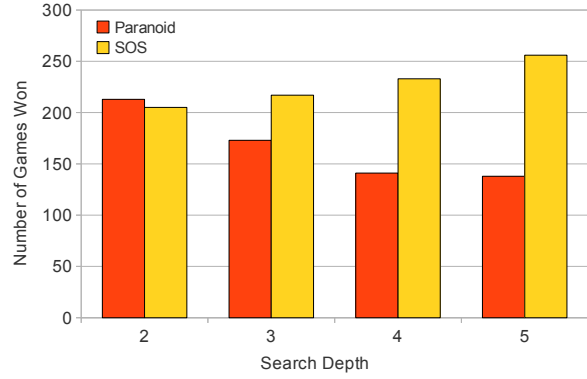


Figure 8: Results of playing Paranoid and our SOS player in 500 Quoridor games against two random-preference players. The SOS player had absolute knowledge of the social-range matrix when playing.

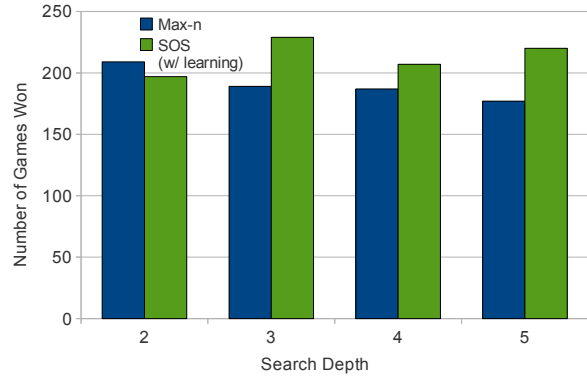


Figure 9: Results of playing Max-n and our SOS player in 500 Quoridor games against two random-preference players. The SOS player learned the social-range matrix with our naive learning algorithm and a history of size 5.

Max-n and Paranoid performance decreases, the paranoid assumption seems to have a greater effect on performance than the rationality assumption of Max-n.

In the last set of experiments we reproduced the same setting as in the second set, but this time used the SOS algorithm with learning, where k was set to 5. That is, the algorithm started each game with a random social matrix and used the suggested learning rule to adapt the values during game play. In figures 9 and 10 we can see that the online learning version of the SOS algorithm also do significantly better than Max-n and Paranoid in most cases ($\mathcal{P} < 0.01$), excluding depth 4 against Max-n, and depth 3 against Paranoid that are not statistically significant.

6. CONCLUSION

In this paper we addressed a fundamental question for search algorithms for extensive-form, multi-player games: *how to describe and reason about inter-player relationships during game play?* Our approach was to model the players' interpersonal relationships by incorporating ideas from *Social Value Orientation* theory into game-tree search.

Our Socially Oriented Search (SOS) algorithm models re-

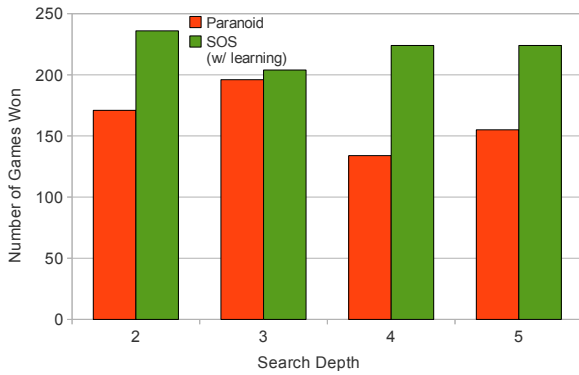


Figure 10: Results of playing Paranoid and our SOS player in 500 Quoridor games against two random-preference players. The SOS player learned the social-range matrix with our naive learning algorithm and a history of size 5.

relationships with a social-range matrix and uses it to compute the *perceived utilities* for each player and guide the search procedure. Our analytical results show that the algorithm can mimic the paranoid and Max-n behaviors by setting the social-matrix elements to specific values, and that the strategy computed by the SOS algorithm is a *perceived equilibrium*. Moreover, our analytical results relate the granularity of the evaluation function to the expected difference in strategies that will be selected by the SOS algorithm and Max-n and Paranoid. This relationship shows that using coarser evaluation functions reduces the ability to recognize social orientations.

We incorporated a simple learning algorithm into SOS to learn the social orientations of the players as the game progresses. The dynamic learning rule uses that the last k actions of each player to adapt its social-matrix during game play. In our evaluations of both the learning and non-learning versions of the algorithm using the Quoridor game, we found that in most cases they produced significantly better play than the Max-n and Paranoid algorithms.

For future work, we plan to evaluate the SOS algorithm in multi-player games that have both probabilistic elements (e.g., Risk) and incomplete information (e.g., Hearts). We also plan to evaluate the learning rule against players that change their orientations during game, as well as experimenting with techniques for learning the value of k .

7. ACKNOWLEDGMENTS

This work was supported in part by DARPA and U.S. Army Research Laboratory contract W911NF-11-C-0037, and AFOSR grant FA95500610405. The information in this paper does not necessarily reflect the position or policy of the U.S. Government, and no official endorsement should be inferred.

8. REFERENCES

[1] W. T. Au and J. Y. yee Kwong. Measurements and effects of social-value orientation in social dilemmas. *Contemporary psychological research on social dilemmas*, pages 71–98, 2004.

[2] R. M. Axelrod. *The evolution of cooperation*. Basic Books, New York, 1984.

[3] Bogaert, Sandy, Boone, Christophe, Declerck, and Carolyn. Social value orientation and cooperation in social dilemmas: A review and conceptual model. *British Journal of Social Psychology*, 47(3):453–480, 2008.

[4] C. G. M. David M. Messick. Motivational bases of choice in experimental games. *Experimental Social Psychology*, 1(4):1–25, 1968.

[5] L. Glendenning. Mastering quoridor - master thesis. Technical report, The University of New Mexico, 2005.

[6] D. Griesinger and J. Livingston. Towards a model of interpersonal motivation in experimental games. *Behavioral Science*, 18:173–188, 1973.

[7] P. Kuznetsov and S. Schmid. Towards Network Games with Social Preferences. *Structural Information and Communication Complexity*, pages 14–28, 2010.

[8] C. Luckhardt and K. Irani. An algorithmic solution of n-person games. In *AAAI*, pages 158–162, 1986.

[9] V. Mamadouh. Grid-group cultural theory: an introduction. *GeoJournal*, 47:395–409, 1999.

[10] S. Markovitch and R. Regeer. Learning and exploiting relative weaknesses of opponent agents. *Autonomous Agents and Multi-Agent Systems*, 10:2005, 2004.

[11] D. Mutchler. The multi-player version of minimax displays game-tree pathology. *Artificial Intelligence*, 64(2):323–336, 1993.

[12] D. S. Nau, M. Luštrek, A. Parker, I. Bratko, and M. Gams. When is it better not to look ahead? *Artificial Intelligence*, 174(16-17):1323–1338, 2010.

[13] J. V. Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.

[14] J. Schaeffer, Y. Björnsson, N. Burch, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen. Solving checkers. In *IJCAI-05*, pages 292–297, 2005.

[15] J. Schaeffer, J. Culberson, N. Treloar, B. Knight, P. Lu, and D. Szafron. A world championship caliber checkers program. *Artificial Intelligence*, 53:53–2, 1992.

[16] N. Sturtevant and R. Korf. On pruning techniques for multi-player games. In *AAAI*, pages 201–208, 2000.

[17] N. R. Sturtevant. A comparison of algorithms for multi-player games. In *CG*, pages 108–122, 2002.

[18] N. R. Sturtevant. Current challenges in multi-player game search. In *Computers and Games*, pages 285–300, 2004.

[19] N. R. Sturtevant, M. Zinkevich, and M. H. Bowling. Prob-maxn: Playing n-player games with opponent models. In *AAAI*, pages 1057–1063, 2006.

[20] I. Zuckerman, A. Felner, and S. Kraus. Mixing search strategies for multi-player games. In *IJCAI*, pages 646–651, 2009.

A Study of Computational and Human Strategies in Revelation Games

Noam Peled
Gonda Brain Research Center
Bar Ilan University, Israel
noam.peled@live.biu.ac.il

Ya'akov (Kobi) Gal
Dept. of Information Systems
Engineering
Ben-Gurion University, Israel
kobig@bgu.ac.il

Sarit Kraus^{*}
Dept. of Computer Science
Bar Ilan University, Israel
sarit@cs.biu.ac.il

ABSTRACT

Revelation games are bilateral bargaining games in which agents may choose to truthfully reveal their private information before engaging in multiple rounds of negotiation. They are analogous to real-world situations in which people need to decide whether to disclose information such as medical records or university transcripts when negotiating over health plans and business transactions. This paper presents an agent-design that is able to negotiate proficiently with people in a revelation game with different dependencies that hold between players. The agent modeled the social factors that affect the players' revelation decisions on people's negotiation behavior. It was empirically shown to outperform people in empirical evaluations as well as agents playing equilibrium strategies. It was also more likely to reach agreement than people or equilibrium agents.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]

General Terms

Experimentation

Keywords

Human-robot/agent interaction, Negotiation

1. INTRODUCTION

In many negotiation settings, participants lack information about each other's preferences, often hindering their ability to reach beneficial agreements. This paper presents a study of a particular class of such settings we call "revelation games". In these settings, players are given the choice to truthfully reveal private information before commencing in a finite sequence of alternating negotiation rounds. Revealing this information narrows the search space of possible agreements and may lead to agreement more quickly, but may also lead players to be exploited by others.

^{*}Sarit Kraus is also affiliated with the University of Maryland Institute for Advanced Computer Studies.

Cite as: A Study of Computational and Human Strategies in Revelation Games, Peled N, Gal Y and Kraus S, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2-6, 2011, Taipei, Taiwan, pp. 345-352.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

Revelation games combine two types of interactions that have been studied in the past: Signaling games [13], in which players choose whether to convey private information to each other, and bargaining [10], in which players engage in multiple negotiation rounds. Revelation games are analogous to real-world scenarios in which parties may choose to truthfully reveal information before negotiation ensues. For example, consider a scenario in which company employees negotiate over the conditions of their employer-sponsored health insurance policy. The employees can waive the right to keep their medical records private. The disclosure of this information to the employer is necessarily truthful and is not associated with a cost to the employees. It may provide employees with favorable conditions when negotiating over future health policies. However, many people choose not to disclose medical records to their employees, fearing they may be compromised by this information.

This paper describes a new agent design that uses a decision-theoretic approach to negotiate proficiently with people in revelation games. The agent explicitly reasons about the social factors that affect people's decisions whether to reveal private information, as well as the effects of people's revelation decisions on their negotiation behavior. It combines a prediction model of people's behavior in the game with a decision-theoretic approach to make optimal decisions. The parameters of this model were estimated from data consisting of human play. The agent was evaluated playing new people and an agent playing equilibrium strategies in a revelation game that varied the dependency relationships between players. The results showed that the agent was able to outperform human players as well as the equilibrium agent. It learned to make offers that were significantly more beneficial to people than the offers made by other people while not compromising its own benefit, and was able to reach agreement significantly more often than did people as well as the equilibrium agent. In particular, it was able to exploit people's tendency to agree to offers that are beneficial to the agent if people revealed information at the onset of the negotiation.

The contributions of this paper are fourfold. First, it formally presents revelation games as a new type of interaction which supports controlled revelation of private information. Second, it presents a model of human behavior that explicitly reasons about the social factors that affect people's negotiation behavior as well as the effects of players' revelation decisions on people's negotiation behavior. Third, it incorporates this model into a decision-making paradigm for an agent that uses the model to make optimal decisions in reve-

lation games. Lastly, it provides an empirical analysis of this agent, showing that the agent is able to outperform people as well as more likely to reach agreement than people.

2. RELATED WORK

Our work is related to studies in AI that use opponent modeling to build agents for repeated negotiation in heterogeneous human-computer settings. These include the KBAgent that made offers with multiple attributes in settings which supported opting out options, and partial agreements [11]. This agent used a social utility function to consider the trade-offs between its own benefit from an offer and the probability that it is accepted by people. It used density estimation to model people’s behavior and approximated people’s reasoning by assuming that people would accept offers from computers that are similar to offers they make to each other. Other works employed Bayesian techniques [6] or approximation heuristics [7] to estimate people’s preferences in negotiation and integrated this model with a pre-defined concession strategy to make offers. Bench-Capon [2] provide an argumentation based mechanism for explaining human behavior in the ultimatum game. We extend these works in two ways, first in developing a partially strategic model of people’s negotiation behavior and second in formalizing an optimal decision-making paradigm for agents using this model. Gal and Pfeffer [4] proposed a model of human reciprocity in a setting consisting of multiple one-shot take-it-or-leave-it games, but did not evaluate a computer agent or show how the model can be used to make decisions in the game. Our work augments these studies in allowing players to reveal private information and in explicitly modeling the effect of revelation on people’s negotiation behavior.

Our work is also related to computational models of argumentation, in that people’s revelation decisions provide an explanation of the type of offers they make during negotiation. Most of these works assume that agents follow pre-defined strategies for revealing information [12, 14] and do not consider or model human participants.

Lastly, revelation games, which incorporate both signaling and bargaining, were inspired by canonical studies showing that people learn to play equilibrium strategies when they need to signal their private information to others [1]. On the other hand, people’s bargaining behavior does not adhere to equilibrium [3, 9], and computers cannot use such strategies to negotiate well with people [8]. Our work shows that integrating opponent modeling and density estimation techniques is an effective approach for creating agents that can outperform people as well equilibrium strategies in revelation games.

3. IMPLEMENTATION: COLORED TRAILS

We based our empirical work on a test-bed called Colored Trails [5], which we adapted to model revelation games with 2 rounds, the minimal number that allows an offer to be made by both players. Our revelation game is played on a board of colored squares. Each player has a square on the board that is designated as its goal. The goal of the game is to reach the goal square. To move to an adjacent square required surrendering a chip in the color of that square. Players had full view of the board and each others’ chips. Both players were shown two possible locations for their goals with associated belief probabilities, but

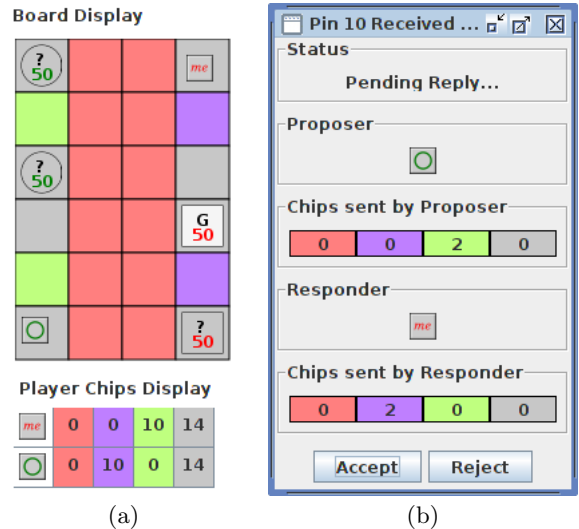


Figure 1: (a) A Colored Trails revelation game shown from Bob’s point of view. (b) Bob’s offer

each player could only see its own goal. An example of a CT revelation game is shown in Figure 1. Here, the “me” and “O” icons represent two players, Bob and Alice, respectively. Each player has two possible goals. Bob’s true goal is located three steps below the “me” icon (appearing as a white G square). Bob’s other goal is located two steps below his true goal (appearing as a grey “?” square). Alice’s possible goals are presented as two grey “?” circles, located three and five steps above Alice’s “O” icon. The board is presented from Bob’s point of view. Bob can see its true goal location but Alice does not observe it. Similarly, Bob cannot observe Alice’s true goal location. The number “50” on each goal square represent a 50% probability that the true goal lies in that square.

Our CT game progresses in three phases with associated time limits. In the revelation phase, both players can choose to truthfully reveal their goal to the other player.¹ In the proposal phase, one of the players is randomly assigned the role of proposer and can offer to exchange a (possibly empty) subset of its chips with a (possibly empty) subset of the chips of the other player. If the responder accepts the offer, the chips are transferred automatically according to the agreement, both participants will automatically be moved as close as possible to the goal square given their chips and the game will end. If the responder rejects (or no offer was received), it will be able to make a counter-proposal. If the proposal is accepted, the game will end with the agreement result as above. Otherwise, the game will end with no agreement.

At the end of the game, the score for each player is computed as follows: 100 points bonus for reaching the goal; 5 points for each chip left in a player’s possession, and 10 points deducted for any square in the path between the players’ final position and the goal-square.² Suppose for example that Alice’s true goal is five steps above the position of her icon (Bob does not see this goal if Alice does not reveal it). Bob is missing one chip to get to the goal while Alice is missing two chips; the score for Alice is 70 points and for

¹This decision is performed simultaneously by all players, and goals are only revealed at the end of the phase.

²This path is computed by the Manhattan distance.

Bob is 90 points.

The game is interesting because players need to reason about the tradeoff between revealing their goals and providing information to the other player, or not to reveal their goals to possibly receive or ask for more chips than they need. In addition, if there is a second round, the proposer in this round has an advantage, in that it makes the final offer in the game. But the identity of the second proposer is not known at the time that players decide whether to reveal their goals.

4. THE SIGAL AGENT

The Sigmoid Acceptance Learning Agent (SIGAL) developed for this study uses a decision-theoretic approach to negotiate in revelation games, that is based on a model of how humans make decisions in the game. Before describing the strategy used by SIGAL we make the following definitions. Each player has a type t_i that represents the true position of its goal on the board.³ Let ω^n represent an offer ω made by a proposer player at round $n \in \{1, 2\}$ in a game. Let $r^n \in \{\text{accept}, \text{reject}\}$ represent the response to ω^n by a responder player. Let s_i represent the score in the game as described in the previous section. The no-negotiation alternative (NNA) score to player i of type t_i is the score for i in the game given that no agreement was reached. We denote the score for this event as $s_i(\emptyset)$.⁴ We denote the benefit to player i from ω^n given that $r^n = \text{accept}$ as $\pi_i(\omega^n | t_i)$. This is defined as the difference in score to i between an offer ω^n and the NNA score:

$$\pi_i(\omega^n | t_i) = s_i(\omega^n | t_i) - s_i(\emptyset) \quad (1)$$

Let T_i denote a set of types for player i . Let ϕ_i denote player i 's decision whether to reveal its type at the onset of the game, which we will refer to as round 0. Let $\Phi_i = t_i^k$ denote the event in which i reveals its type $t_i^k \in T_i$, and let $\Phi_i = \text{null}$ denote the event in which i does not reveal its type. Let h^n denote a history of moves, including for both players i and j their revelation decision at the onset of the game, and the proposals and responses for rounds 1 through n . We define h^0 and h^1 as follows:

$$h^0 = \{\phi_i, \phi_j\}; h^1 = \{h^0, \omega^1, r^1\} \quad (2)$$

For the remainder of this section, we assume that the SIGAL agent (denoted a) is paying a person (denoted p). Let $\omega_{a,p}^n$ represent an offer made by the agent to the person in round n and let r_p^n represent the response of the person to $\omega_{a,p}^n$. The expected benefit to SIGAL from $\omega_{a,p}^n$ given history h^{n-1} and SIGAL's type t_p is denoted $E_a(\omega_{a,p}^n | h^{n-1}, t_a)$. Let $p(r_p^n = \text{accept} | \omega_{a,p}^n, h^{n-1})$ denote the probability that $\omega_{a,p}^n$ is accepted by the person given history h^{n-1} .

We now specify the strategy of SIGAL for the revelation game defined in Section 3. The strategy assumes there exists a model of how humans make and accept offers in both rounds. We describe how to estimate the parameters of this model in Section 5. We begin by describing the negotiation strategies of SIGAL for rounds 2 and 1.

³Revealing goals in the game thus corresponds to making types common knowledge.

⁴Note that if no agreement was reached in round 2 (the last round) of the game, players' NNA score is also their final score in the game. If no agreement was reached in round 1 of the game, players' final score depends on whether the counter-proposal in round 2 is accepted.

Round 2: If SIGAL is the second proposer, its expected benefit from an offer ($\omega_{a,p}^2$) depends on its model of how people accept offers in round 2, encapsulated in the probability $p(r_p^2 = \text{accept} | \omega_{a,p}^2, h^1)$. The benefit to SIGAL is

$$\begin{aligned} E_a(\omega_{a,p}^2 | h^1, t_a) = & \\ & \pi_a(\omega_{a,p}^2 | t_a) \cdot p(r_p^2 = \text{accept} | \omega_{a,p}^2, h^1) + \\ & \pi_a(\emptyset | t_a) \cdot p(r_p^2 = \text{reject} | \omega_{a,p}^2, h^1) \end{aligned} \quad (3)$$

Here, the term $\pi_a(\emptyset | t_a)$ represents the benefit to SIGAL from the NNA score, which is zero. SIGAL will propose an offer that maximizes its expected benefit in round 2 out of all possible proposals for this round.

$$\omega_{a,p}^{2*} = \operatorname{argmax}_{\omega_{a,p}^2} E_a(\omega_{a,p}^2 | h^1, t_a) \quad (4)$$

If SIGAL is the second responder, its optimal action is to accept any proposal from the person that gives it positive benefit. Let $r_a^{2*}(\omega_{p,a}^2 | h^1)$ denote the response of SIGAL to offer $\omega_{p,a}^2$, defined as

$$r_a^{2*}(\omega_{p,a}^2 | h^1) = \begin{cases} \text{accept} & \pi_a(\omega_{p,a}^2 | t_a) > 0 \\ \text{reject} & \text{otherwise} \end{cases} \quad (5)$$

where $\pi_a(\omega_{p,a}^2 | t_a)$ is defined in Equation 1. The benefit to SIGAL from this response is defined as

$$\begin{aligned} \pi_a(r_a^{2*} | \omega_{p,a}^2, h^1, t_a) = & \\ & \begin{cases} \pi_a(\omega_{p,a}^2 | t_a) & r_a^{2*}(\omega_{p,a}^2 | h^1) = \text{accept} \\ \pi_a(\emptyset | t_a) & \text{otherwise} \end{cases} \end{aligned} \quad (6)$$

Round 1: If SIGAL is the first proposer, its expected benefit from making a proposal $\omega_{a,p}^1$ depends on its model of the person: If the person accepts $\omega_{a,p}^1$, then the benefit to SIGAL is just $\pi_a(\omega_{a,p}^1 | t_a)$. If ($\omega_{a,p}^1$) is rejected by the person, then the benefit to SIGAL depends on the counter-proposal $\omega_{p,a}^2$ made by the person in round 2, which itself depends on SIGAL's model $p(\omega_{p,a}^2 | h^1)$ of how people make counter-proposals. The expected benefit to SIGAL from behaving optimally as a second responder for a given offer $\omega_{p,a}^2$ is denoted $E_a(\text{resp}^2 | h^1, t_a)$, and defined as

$$\begin{aligned} E_a(\text{resp}^2 | h^1, t_a) = & \\ & \sum_{\omega_{p,a}^2} p(\omega_{p,a}^2 | h^1) \cdot \pi_a(r_a^{2*} | \omega_{p,a}^2, h^1, t_a) \end{aligned} \quad (7)$$

where $\pi_a(r_a^{2*} | \omega_{p,a}^2, h^1, t_a)$ is defined in Equation 6.

Its expected benefit from $\omega_{a,p}^1$ is:

$$\begin{aligned} E_a(\omega_{a,p}^1 | h^0, t_a) = & \\ & \pi_a(\omega_{a,p}^1 | t_a) \cdot p(r_p^1 = \text{accept} | \omega_{a,p}^1, h^0) + \\ & E_a(\text{resp}^2 | h^1, t_a) \cdot p(r_p^1 = \text{reject} | \omega_{a,p}^1, h^0) \end{aligned} \quad (8)$$

Where $h^1 = \{h^0, \omega_{a,p}^1, r_p^1 = \text{reject}\}$. SIGAL will propose an offer in round 1 that maximizes its expected benefit in this round:

$$\omega_{a,p}^{1*} = \operatorname{argmax}_{\omega_{a,p}^1} E_a(\omega_{a,p}^1 | h^0, t_a) \quad (9)$$

If SIGAL is the first responder, it accepts any offer that provides it with a larger benefit than it would get from making the counter-proposal $\omega_{a,p}^{2*}$ in round 2, given its model of

how people respond to offers in round 2:

$$r_a^{1*}(\omega_{p,a}^1 | h^0) = \begin{cases} \text{accept} & \pi_a(\omega_{p,a}^1 | t_a) > \\ & E_a(\omega_{a,p}^{2*} | h^1, t_a) \\ \text{reject} & \text{otherwise} \end{cases} \quad (10)$$

Here, $h^1 = \{h^0, \omega_{p,a}^1, r_a^1 = \text{reject}\}$, $\pi_a(\omega_{p,a}^1 | t_a)$ is defined in Equation 1 and $E_a(\omega_{a,p}^{2*} | h^1, t_a)$ is the benefit to SIGAL from making an optimal proposal $\omega_{a,p}^{2*}$ at round 2, as defined in Equation 3.

Let $\pi_a(r_a^{1*} | \omega_{p,a}^1, h^0, t_a)$ denote the benefit to SIGAL from its response to offer $\omega_{p,a}^1$ in round 1. If SIGAL accepts this offer, it receives the benefit associated with $\omega_{p,a}^1$. If it rejects this offer, it will receive the expected benefit $E_a(\omega_{a,p}^{2*} | h^1, t_a)$ from making an optimal counter-proposal at round 2:

$$\pi_a(r_a^{1*} | \omega_{p,a}^1, h^0, t_a) = \begin{cases} \pi_a(\omega_{p,a}^1 | t_a) & r_a^{1*}(\omega_{p,a}^1 | h^0) = \text{accept} \\ E_a(\omega_{a,p}^{2*} | h^1, t_a) & \text{otherwise} \end{cases} \quad (11)$$

The expected benefit to SIGAL as a responder in round 1 is denoted as $E_a(\text{resp}^1 | h^0, t_a)$. This benefit depends on its model of all possible offers made by people for each type, given that SIGAL responds optimally to the offer.

$$E_a(\text{resp}^1 | h^0, t_a) = \sum_{t_p \in T_p} p(t_p | h^0) \cdot \left(\sum_{\omega_{p,a}^1} p(\omega_{p,a}^1 | t_p, h^0) \cdot \pi_a(r_a^{1*} | \omega_{p,a}^1, h^0, t_a) \right) \quad (12)$$

Note that when the person reveals his/her type at round 0, this is encapsulated in the history h^0 , and $p(t_p | h^0)$ equals 1 for the person's true type. Otherwise $p(t_p | h^0)$ equals the probability $p(t_p)$.

Round 0: In the revelation round SIGAL needs to decide whether to reveal its type. Let $E_a(h^0, t_a)$ denote the expected benefit to SIGAL given that h^0 includes a revelation decision for both players and that t_a is the type of agent. This benefit depends on the probability that SIGAL is chosen to be a proposer ($p(\text{prop})$) or responder ($p(\text{resp})$) in round 1:

$$E_a(h^0, t_a) = p(\text{resp}) \cdot E_a(\text{resp}^1 | h^0, t_a) + p(\text{prop}) \cdot E_a(\omega_{a,p}^{1*} | h^0, t_a) \quad (13)$$

Here, $\omega_{a,p}^{1*}$ is the optimal proposal for SIGAL in round 1, and $E_a(\omega_{a,p}^{1*} | h^0, t_a)$ is the expected benefit associated with this proposal, defined in Equation 8.

Because players do not observe each other's revelation decisions, the expected benefit for a revelation decision ϕ_a of the SIGAL agent sums over the case where people revealed their type (i.e., $\phi_a = t_p$) or did not reveal their type (i.e., $\phi_a = \text{null}$). We denote $p(\phi_p = t_p)$ as the probability that the person revealed its type t_p , and $p(\phi_p = \text{null})$ as the probability that the person did not reveal its type t_p .

$$E_a(\phi_a) = \sum_{t_p \in T_p} [p(\phi_p = t_p) \cdot E_a(h^0 = \{\phi_a, \phi_p = t_p\}, t_a) + p(\phi_p = \text{null}) \cdot E_a(h^0 = \{\phi_a, \phi_p = \text{null}\}, t_a)] \quad (14)$$

Given that SIGAL is of type $t_a \in T_a$, it reveals its type only if its expected benefit from revelation is strictly greater from not revealing:

$$\phi_a^* = \begin{cases} t_a & E_a(\phi_a = t_a) \geq \\ & E_a(\phi_a = \text{null}) \\ \text{null} & \text{otherwise} \end{cases} \quad (15)$$

The value of the game for SIGAL for making the optimal decision whether to reveal its type is defined as $E_a(\phi_a^*)$.

Lastly, we wished SIGAL to take a risk averse approach to making decisions in the game. Therefore SIGAL used a convex function to represent its utility in the game from an offer ω^n , which modified Equation 1.

$$\pi'_a(\omega^n | t_a) = \frac{\pi_a(\omega^n | t_a)^{(1-\rho)}}{1-\rho} \quad (16)$$

The strategy used by SIGAL is obtained by "plugging in" the risk averse utility $\pi'_a(\omega^n | t_a)$ instead of $\pi_i(\omega^n | t_i)$.

5. MODELING HUMAN PLAYERS

In this section we describe a model of people's behavior used by SIGAL to make optimal decisions in the game. We assume that there is a training set of games played by people, as we show in the next Section.

5.1 Accepting Proposals

We modeled people's acceptance of proposals in revelation games using a stochastic model that depended on a set of features. These comprised past actions in the game (e.g., a responder may be more likely to accept a given offer if it revealed its type as compared to the case in which it did not reveal its type) as well as social factors (e.g., a responder player may be less likely to accept a proposal that offers more benefit to the proposer than to itself).⁵

Let $\omega_{i,j}^n$ represent a proposal from a player i to a player j at a round n . We describe the following features that affect the extent to which player j will accept proposal $\omega_{i,j}^n$. These features are presented from the point of view of proposer i , therefore we assume that the type of the proposer t_i is known, while the type of the responder t_j is known only if j revealed its type. We first detail the features that relate to players' decisions whether to reveal their types.

- REV_j^0 . Revelation by j . This feature equals 1 if the responder j has revealed its type and 0 otherwise. The superscript 0 indicates this feature is relevant to the revelation phase, which is round 0.
- REV_i^0 . Revelation by i . This feature equals 1 if the proposer has revealed its type t_i .

We now describe the set of features relating to social factors of the responder player j .

- BEN_j^n . Benefit to j . The benefit to j from proposal $\omega_{i,j}^n$ in round n . This measures the extent to which the proposal $\omega_{i,j}^n$ is generous to the responder. In the case where j revealed its type, this feature equals $\pi_j(\omega_{i,j}^n | t_j)$ and computed directly from Equation 1. Otherwise, the value of this feature is the expected

⁵Both of these patterns were confirmed empirically, as shown in the Results section.

benefit to the responder from $\omega_{i,j}^n$ for all possible responder types T_j :

$$\sum_{t_j \in T_j} p(t_j | h^{n-1}) \cdot \pi_j(\omega_{i,j}^n | t_j)$$

- AI_i^n . Advantageous inequality of i . The difference between the benefit to proposer i and responder j that is associated with proposal $\omega_{i,j}^n$. This measures the extent to which proposer i is competitive, in that $\omega_{i,j}^n$ offers more for i than for j . This feature equals the difference between $\pi_i(\omega_{i,j}^n, \text{accept} | t_i)$ and BEN_j^n .

To capture the way the behavior in round $n = 1$ affects the decisions made by participants in round $n = 2$, we added the following features that refer to past offers.

- $P.BEN_j^n$. Benefit to j in the previous round. This feature equals BEN_j^1 if $n = 2$, and 0 otherwise.
- $P.BEN_i^n$. Benefit to proposer i in the previous round. This feature equals $\pi_i(\omega_{i,j}^1, \text{accept} | t_i)$ if $n = 2$ and 0 otherwise.

To illustrate, consider the CT board game shown in Figure 1. Alice is missing two green chips to get to the goal and Bob is missing 1 purple chip to get to the goal. Suppose Bob is the first proposer (player i) and that Alice is the first responder (player j), and that Bob revealed its goal to Alice, so its type is common knowledge, while Alice did not reveal her goal. We thus have that $REV_j^0 = 0$ and $REV_i^0 = 1$. Alice’s no-negotiation alternative (NNA) score, $s_j(\emptyset)$, is 70 points and Bob’s NNA score is 90 points.

According to the offer shown in the Figure, Bob offered two green chips to Alice in return for two purple chips. If accepted, this offer would allow Alice to get to the goal in 5 steps, so she will have 19 chips left at the end of the game, worth $19 \cdot 5 = 95$ points. Similarly, Bob will have 21 chips left at the end of the game, worth 105 points. Both will also earn a bonus of 100 points for getting to the goal. Therefore we have that $BEN_j^1 = 95 + 100 - 70 = 125$. Similarly, Bob’s benefit from this proposal is $105 + 100 - 90 = 115$ points. The difference between the benefit to Bob and to Alice is -10 , so we have that $AI_i^1 = -10$. Lastly, because the offer is made in round 1, we have that $P.BEN_j^1 = P.BEN_i^1 = 0$. This offer is more generous to Alice than it is to Bob.

Suppose now that Alice rejects this offer and makes a counter proposal in round 2, that proposes one purple chip to Bob in return for four greens. In this example, Alice is using her knowledge of Bob’s type to make the minimal offer that would allow Bob to reach the goal while providing additional benefit to Alice. Alice is the proposer (player i) and Bob is the responder (player j). Recall that Bob has revealed its goal while Alice did not, so we have $REV_j^0 = 1$ and $REV_i^0 = 0$. Using a similar computation from before, we get that Bob’s score from the counter proposal is 190 points. Therefore we have that $BEN_j^2 = 190 - 90 = 100$. Alice’s benefit from the counter-proposal is $210 - 70 = 140$, therefore we have that $AI_i^2 = 140 - 100 = 40$. The last features in the example capture the benefit to both players from the proposal made in the first round to Alice and Bob, so we have $P.BEN_j^2 = 125$, and $P.BEN_i^2 = 115$.

5.1.1 Social Utility Function

We model the person as using a social utility function to decide whether to accept proposals in the game. This social

utility depends on a weighted average of the features defined above. We define a transition function, T^n , that maps an offer ω^n and history h^{n-1} to an (ordered) set of feature values x^n as follows.⁶

$$x^n = (REV_j^0, REV_i^0, BEN_j^n, AI_i^n, P.BEN_j^n, P.BEN_i^n)$$

To illustrate, in the example above, we have that $x^1 = (0, 1, 125, -10, 0, 0)$ and $x^2 = (1, 0, 100, 40, 125, 115)$.

Let $u(x^n)$ denote the social utility function which is defined as the weighted sum of these features. To capture the fact that a decision might be implemented noisily, we use a sigmoid function to describe the probability that people accept offers, in a similar way to past studies for modeling human behavior [4]. We define the probability of acceptance for a particular features values x^n by a responder to be

$$p(r_i^n = \text{accept} | \omega^n, h^{n-1}) = \frac{1}{1 + e^{-u(x^n)}} \quad (17)$$

where $x^n = T^n(\omega^n, h^{n-1})$. In particular, the probability of acceptance converges to 1 as $u(x^n)$ becomes large and positive, and to 0 as the utility becomes large and negative. We interpret the utility to be the degree to which one decision is preferred. Thus, the probability of accepting a proposal is higher when the utility is larger.

5.1.2 Estimating Weights

To predict how people respond to offers in the game, it is needed to estimate the weights in their social utility function in a way that best explains the observed data. In general, we need to model the probability that an offer is accepted for any possible instantiation of the history. The number of possible proposals in round 1 is exponential in the combined chip set of players.⁷ It is not possible to use standard density estimation techniques because many such offers were not seen in the training set or were very rare. Therefore, we employed a supervised learning approach that assumed people used a noisy utility function to accept offers that depended on the features defined above. Let $\Omega_{i,p}$ denote a data set of offers proposed by some participant i to a person p .⁸ For each offer $\omega_{i,p}^n \in \Omega_{i,p}$ let $y(r_p^n | \omega_{i,p}^n)$ denote an indicator function that equals 1 if the person accepted proposal $\omega_{i,p}^n$, and zero otherwise. The error of the predictor depends on the difference between $y(r_p^n | \omega_{i,p}^n)$ and the predicted response $p(r_p^n = \text{accept} | \omega_{i,p}^n, h^{n-1})$, as follows:

$$\sum_{\omega_{i,p}^n \in \Omega_{i,p}} (p(r_p^n = \text{accept} | \omega_{i,p}^n, h^{n-1}) - y(r_p^n | \omega_{i,p}^n))^2 \quad (18)$$

where $p(r_p^n = \text{accept} | \omega_{i,p}^n, h^{n-1})$ is defined in Equation 17.

We used a standard Genetic algorithm to estimate weight values for the features of people’s social utility that minimize the aggregate error in the training set. To avoid overfitting the training set, we used a held-out cross-validation set consisting of 30% of the data. We chose the instance with minimal error (on the training set) in the generation that corresponded to the smallest error on the cross-validation

⁶These weights are estimated from data using statistical techniques as described in the following section.

⁷In one of the boards we studied the number of possible offers that provided the same benefit to both players was about 27,000, out of a total of 2^{24} possible offers.

⁸We explain how we collected this data set in the Empirical Methodology Section.

set. We used ten-fold cross-validation, repeating this process ten times, each time choosing different training and testing sets, producing ten candidate instances. To pick the best instance, we computed the value of the game $E_a(\phi_a^*)$ for SIGAL for each of the learned models, where ϕ_a^* is defined in Equation 15. This is the expected benefit for SIGAL given that it chooses optimal actions using a model of people that corresponds to the feature values in each instance.

5.2 Proposing and Revealing

This section describes our model of how people make proposals in revelation games and reason about whether to reveal information.

5.2.1 First proposal model

We used standard density estimation techniques (histograms) to predict people’s offers for different types. Based on the assumption that proposals for the first round depend on the proposer’s type and its decision whether to reveal, we divided the possible proposals to equivalence classes according to the potential benefit for the proposer player, and counted how many times each class appears in the set. Let $p(\omega_{p,j}^1 | t_p, \phi_i)$ denote the probability that a human proposer of type t_p offers $\omega_{p,j}^1$ in round 1. Let $N_{t_p, \phi_p}(\pi_p(\omega_{p,j}^1 | t_p))$ denote the number of first proposals which gives the human a benefit of $\pi_p(\omega_{p,j}^1 | t_p)$, given the human is of type t_p and its revelation decision was ϕ_p . Let $N_{t_p, \phi_p}(\Omega_{p,j}^1)$ denote the number of the first proposal in this subset. $p(\omega_{p,j}^1 | t_p, \phi_p)$ is defined as:

$$p(\omega_{p,j}^1 | t_p, \phi_p) = \frac{N_{t_p, \phi_p}(\pi_p(\omega_{p,j}^1 | t_p))}{N_{t_p, \phi_p}(\Omega_{p,j}^1)} \quad (19)$$

5.2.2 Counter-proposal model

According to our model, a player’s proposal in the second round also depends on the history, this two dimensional probability density function tends to be too much sparse to calculate it directly as described in Subsection 5.2.1. Inspired by studies showing that people engage in tit-for-tat reasoning [15] we used this principal to model the counter-proposals made by people. We assumed that a responder player i will be proposed offer $\omega_{p,i}^2$ by a human player in the second round with benefit $\pi_i(\omega_{p,i}^2 | t_i)$ that is equal to the benefit $\pi_p(\omega_{i,p}^1 | t_p)$ from offer $\omega_{i,p}^1$ made to the people in the first round, when the human was a responder. For example, suppose that Bob is the proposer in round 1 and propose to Alice a benefit of 125. According to the model, if Alice rejects the offer she will propose Bob a counter-proposal that provides Bob with the same benefit, 125. Note that this does not assume that the proposal will provide Alice with the same benefit she got from Bob in the proposal from round 1. Formally, let $N_{\Omega_{p,i}^2}(\pi_p(\omega_{i,p}^1 | t_p))$ denote the number of counter-proposals $\omega_{p,i}^2$ which give benefit $\pi_p(\omega_{i,p}^1 | t_p)$. We assume that there always exists at least one proposal that meets this criterion, i.e., $N_{\Omega_{p,i}^2}(\pi_p(\omega_{i,p}^1 | t_p)) \neq 0$. The “tit for tat” heuristic is as follows:

$$p(\omega_{p,i}^2 | h^1) = \begin{cases} 0 & \pi_i(\omega_{p,i}^2) \neq \pi_p(\omega_{i,p}^1) \\ 1/N_{\Omega_{p,i}^2}(\pi_p(\omega_{i,p}^1 | t_p)) & \text{otherwise} \end{cases} \quad (20)$$

This heuristic is used in Equation 7 to facilitate the computation of the expected benefit from SIGAL as a responder in round 1.

Lastly, we detail the model used by SIGAL to predict whether the person reveals its goal. Let N_{t_p} denote the number of instances in which people were of type t_p , and let $N_{t_p}(\phi_p)$ denote the number of times that people of type t_p chose to reveal their type. The probability that a human player p revealed its type t_p is defined as:

$$p(\phi_p | t_p) = \frac{N_{t_p}(\phi_p)}{N_{t_p}} \quad (21)$$

6. EMPIRICAL METHODOLOGY

In this section we describe the methodology we used in order to learn the parameters of the model of how people play revelation games, and to evaluate it. For these purposes we recruited 228 students enrolled in a computer science or software engineering program at several universities and colleges. Subjects received an identical tutorial on revelation games that was exemplified on a board (not the boards used in the study). Actual participation was contingent on successfully answering a set of basic comprehension questions about the game. Participants were seated in front of a terminal for the duration of the study, and could not speak to any of the other participants. Each participant played two revelation games on different boards. The boards in the study fulfilled the following conditions at the onset of the game: (1) There were two goals for each player; (2) Every player lacked one or two chips to reach each of its possible goals; (3) Every player possessed the chips that the other needed to get to each of its possible goals; (4) There existed at least one exchange of chips which allowed both players to reach each of their possible goals; (5) the goals were distributed with a probability of 50% for both players. We used two boards in the study. In the “asymmetric board”, one of the players needed a single chip of a particular color to reach its goal, while the other player needed two chips of a particular color to reach its respective goal. This is the board that is shown in Figure 1. We also used a “symmetric board” in which both players needed a single chip of one of two possible colors to get to their goal.

Participants played both symmetric and asymmetric boards in random order. They engaged in a neutral activity (answering demographic questions) between games to minimize the effects of their behavior in the first game on their behavior in the second game. The participant chosen to be the proposer in the first game was randomly determined, and participants switched roles in the second game, such that the proposer in the first game was designated as the responder in the second game. A central server (randomly) matched each participant with a human or an agent counterpart for each game. The identity of each participant was not disclosed. We collected the board layout, and players’ proposals, responses and revelation decisions for all of the games played. To avoid deception all participants were told they would be interacting with a computer or a person. Participants received fixed compensation (course credit) for participating in the experiment.⁹

We divided subjects into four pools. The first pool consisted of people playing other people (66 games). The second pool consisted of people playing a computer agent that used a randomized strategy to make offers and responses (170

⁹Our goal was to build an agent that negotiates well with people, not to explain people’s incentives, therefore fixed compensation was sufficient.

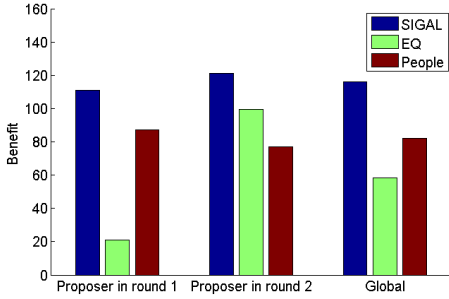


Figure 2: Performance comparison

games). The purpose for this pool was to collect people’s actions for diverse situations, for example, their response to offers that were never made by other people. Two thirds (44 games) of the data from the first pool and the data from the second pool were used for training a model of people’s behavior. The third pool consisted of people playing the SIGAL agent (110 games). The fourth pool (118 games) consisted of people playing an agent using an equilibrium strategy to play revelation games.

7. RESULTS AND DISCUSSION

The performance of SIGAL was measured by comparing its performance against people in the third pool with people’s play in the remaining third of the first pool.¹⁰ We list the number of observations and means for each result. All results reported in this section are statistically significant in the $p < 0.05$ range.

7.1 Analysis: General Performance

We first present a comparison of the performance of SIGAL and people. Figure 2 shows the average benefit (the difference in score between agreement and the no-negotiation alternative score) for different roles (proposers and responder). As shown by the figure, the SIGAL agent outperformed people in all roles (111 points as proposer in round 1 versus 87 points for human proposers in round 1; 121 points as proposer in round 2 versus 77 points for human proposers in round 2).

The SIGAL agent was also more successful at reaching agreements than were people. Only 2% of games in which SIGAL played people did not reach agreement (in first or second round), while 27% of games in which people played other people did not reach agreement. In particular, offers made by SIGAL in round 2 were accepted 87% of the time, while offers made by people in round 2 were only accepted 14% of the time. If an offer is rejected at this last round, the game ends without agreement. This striking difference shows that SIGAL learned to make good offers at critical points in the game.

As shown in Figure 2 SIGAL also outperformed the equilibrium agent in both rounds. The equilibrium agent was fully strategic and assumed the other player was unboundedly rational. Although not shown in the Figure, it made very selfish offers in the last round, offering only 25 average points to people and 215 to itself. Most of these offers were

¹⁰Although this portion corresponds to only 22 games played by people, it was sufficient to achieve statistical significance.

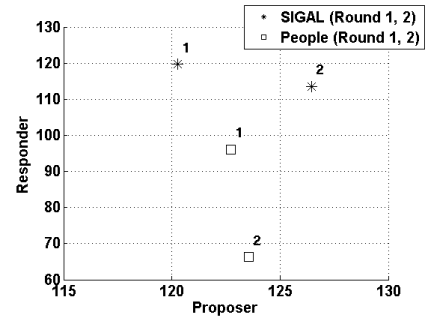


Figure 3: Average Proposed Benefit in First and Second rounds

not accepted. In the first round, it made offers that were highly beneficial to people, offering 219 average points to people and 20 to itself. Most of these offers were accepted, but this did not aid its performance.

To explain the success behind SIGAL’s strategy, we present a comparison of the benefit from proposals made by the SIGAL agent and people in both game rounds in Figure 3. As shown by the Figure both people and SIGAL made offers that were beneficial to both players in rounds 1 and 2. However, SIGAL made offers that were significantly more generous to human responders than did human proposers (120 points benefit provided by SIGAL as proposer in round 1 versus 96 points provided by human proposers; 114 points benefit provided by SIGAL as proposer in round 2 versus 66 points provided by human proposers). As shown by the figure, there was no significant differences between the benefit to SIGAL from offers made by SIGAL itself and people (121 points to SIGAL versus 123 points to people for round 1 and 126 points versus 124 points in round 2). In particular, all of SIGAL’s proposals enabled the responder to reach its goal. Thus, SIGAL was able to learn to make offers that were better for human responders without compromising its own utility.

SIGAL’s strategy is highlighted by examining the weights learned for the different features of how people accept offers. As shown in Table 1, the largest weight was assigned to BEN_j^p , the benefit to the responder from an offer. In addition, the weight for AI_i^p measuring the difference between the benefit for the proposer and responder was large and negative. This means that responders prefer proposals that provide them with large benefits, and are also competitive, in that they dislike offers that provide more to proposers than to responders. The offers made by SIGAL reflect these criteria. In particular, proposers asked more for themselves than for responders in both rounds. In contrast, SIGAL equalized the difference in benefit between proposers and responders in round 1, and decreased the difference between its own benefit and responder’s benefit in round 2 as compared to human proposer.

7.2 Analysis: Revelation of Goals

We now turn to analyzing the affect of goal revelation on the behavior of SIGAL. Recall that $E_a(\phi_a^* = t_a)$ denotes the value of the game for SIGAL when deciding to reveal its goal in round 0, and behaving optimally according to its model of how people make offers. Similarly, $E_a(\phi_a^* = null)$ denotes the value of the game for SIGAL when deciding not

Feature	Value
REV_j^0	0.258
REV_i^0	0.035
BEN_j^n	0.956
AI_i^n	-0.792
$P.BEN_j^n$	0.496
$P.BEN_i^n$	0.334
Free Parameter	0.608

Table 1: Features coefficients weights

to reveal its goal in round 0. Our model predicted no significant difference in value to SIGAL between revealing and not revealing its goal, i.e. $E_a(\phi_a^* = null) \approx E_a(\phi_a^* = t_a)$ for each type $t_a \in T_a$. Therefore we used two types of SIGAL agents, one that consistently revealed its goal at the onset of the game and one that did not reveal. In all other respects these agents followed the model described in Section 4. The empirical results confirmed the model’s prediction, in that there was no significant difference in the performance of the two SIGAL agents for all boards and types used in the empirical study. The results described in this section average over the revealing and non-revealing types of SIGAL agents.

This was confirmed by the empirical results, in which the average performance of the SIGAL agent when revealing its goal was 114 points ($n = 52$), while the average performance of SIGAL when not revealing its goal was 118 points ($n = 58$). This difference was not significantly significant in the $p < 0.05$ range.

However, the decision of the person to reveal or not reveal its goal had a significant affect on the negotiation strategy of SIGAL. When people revealed their goals, SIGAL learned to ask for more benefit for itself as compared to the case in which people did not reveal their goals. For example, when playing the asymmetric board, the non-revealing SIGAL agents learns to ask 125 points for itself if the person reveals its goal, and only 115 points for itself if the person did not reveal. In this case SIGAL took advantage of the fact that the type of the human responder is known, but its own type is not known.

Lastly, the probabilities that people revealed their goals, as learned from the training set, were as follows: 37.14% and 46.27% in the asymmetric board were missing one, and two chips to get to the goal, respectively, and 41.13% for the symmetric board, in which both players were only missing one chip. Interestingly, people missing two chips to get to the goal were most likely to reveal their type. We hypothesize this was to justify their request for their missing chips from the other player.

8. CONCLUSION AND FUTURE WORK

This paper presented an agent-design for interacting with people in “revelation games”, in which participants are given the choice to truthfully reveal private information prior to negotiation. The decision-making model used by the agent reasoned about the social factors that affect people’s decisions whether to reveal their goals, as well as the effects of people’s revelation decisions on their negotiation behavior. The parameters of the model were estimated from data consisting of people’s interaction with other people. In empirical investigations, the agent was able to outperform people playing other people as well as agents playing equilibrium strategies and was able to reach agreement significantly more often than did people.

We are currently extending this work in two directions. First, we are considering more elaborate settings in which players are able to control the extent to which they reveal their goals. Second, we are using this work as the basis for a more broad argumentation in which agents integrate explanations and justifications within their negotiation process.

9. ACKNOWLEDGMENTS

This research was partially funded by European Union Seventh Framework Programme agreement no. PIRG07-GA-2010-268362, by the U.S. Army Research Laboratory and the U.S. Army Research Office under grant number W911NF-08-1-0144 and by NSF grant 0705587. Thanks to Yael Blumberg and Ofra Amir for programming efforts and useful comments.

10. REFERENCES

- [1] J. Banks, C. F. Camerer, and D. Porter. Experimental tests of nash refinements in signaling games. *Games and Economic Behavior*, 6:1–31, 1994.
- [2] T. Bench-Capon, K. Atkinson, and P. McBurney. Altruism and agents: an argumentation based approach to designing agent decision mechanisms. In *Proc. of AAMAS*, 2009.
- [3] I. Erev and A. E. Roth. Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria. *American Economic Review*, 88(4):848–881, 1998.
- [4] Y. Gal and A. Pfeffer. Modeling reciprocity in human bilateral negotiation. In *Proc. of AAAI*, 2007.
- [5] B. Grosz, S. Kraus, S. Talman, and B. Stossel. The influence of social dependencies on Decision-Making. In *Proc. of AAMAS*, 2004.
- [6] K. Hindriks and D. Tykhonov. Opponent modelling in automated multi-issue negotiation using bayesian learning. In *Proc. of AAMAS*, pages 331–338, 2008.
- [7] C. M. Jonker, V. Robu, and J. Treur. An agent architecture for multi-attribute negotiation using incomplete preference information. *Autonomous Agents and Multi-Agent Systems*, 15(2):221–252, 2007.
- [8] R. Lin and S. Kraus. Can automated agents proficiently negotiate with humans? *Communications of the ACM*, 53(1):78–88, 2010.
- [9] R. D. McKelvey and T. R. Palfrey. An experimental study of the centipede game. *Econometrica: Journal of the Econometric Society*, 60(4):803–836, 1992.
- [10] M. J. Osborne and A. Rubinstein. *A course in game theory*. MIT press, 1999.
- [11] Y. Oshrat, R. Lin, and S. Kraus. Facing the challenge of human-agent negotiations via effective general opponent modeling. In *Proc. of AAMAS*, 2009.
- [12] P. Pasquier, R. Hollands, F. Dignum, I. Rahwan, and L. Sonenberg. An empirical study of interest-based negotiation. In *Proc. of ICEC*, 2007.
- [13] A. M. Spence. *Market signaling*. Harvard Univ. Press, 1974.
- [14] K. Sycara. Persuasive argumentation in negotiation. *Theory and decision*, 28(3):203–242, 1990.
- [15] C. Wedekind and M. Milinski. Human cooperation in the simultaneous and the alternating prisoner’s dilemma. *PNAS*, 93(7):2686, 1996.

Efficient Heuristic Approach to Dominance Testing in CP-nets

Minyi Li
Swinburne University of
Technology
myli@swin.edu.au

Quoc Bao Vo
Swinburne University of
Technology
BVO@swin.edu.au

Ryszard Kowalczyk
Swinburne University of
Technology
RKowalczyk@swin.edu.au

ABSTRACT

CP-net (Conditional Preference Network) is one of the extensively studied languages for representing and reasoning with preferences. The fundamental operation of dominance testing in CP-nets, i.e. determining whether an outcome is preferred to another, is very important in many real-world applications. Current techniques for solving general dominance queries is to search for *improving flipping sequence* from one outcome to another as a proof of the dominance relation in all rankings satisfying the given CP-net. However, it is generally a hard problem even for binary-valued, acyclic CP-nets and tractable search algorithms exist only for specific problem classes. Hence, there is a need for efficient algorithms and techniques for dominance testing in more general problem settings. In this paper, we propose a heuristic approach, called DT^* , to dominance testing in arbitrary acyclic multi-valued CP-nets. Our proposed approach guides the search process efficiently and allows significant reduction of search effort without impacting soundness or completeness of the search process. We present results of experiments that demonstrate the computational efficiency and feasibility of our approach to dominance testing.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent Systems

General Terms

Algorithms, Design

Keywords

CP-nets; Dominance Testing; Heuristic

1. INTRODUCTION

The problems of eliciting, representing and reasoning with qualitative preferences over multi-attribute domain arise in many fields such as planning, design, and collective decision making [5, 6, 7, 8]. As the number of alternative outcomes of such domains is exponentially large in the number of attributes, it is unpractical to express preferences explicitly by giving out the ordering over the alternative outcome space. Therefore, the AI research community has developed languages for representing preferences in such domains in a succinct way, exploiting structural properties such as conditional preferential independence. The formalism of CP-nets

Cite as: Efficient Heuristic Approach to Dominance Testing in CP-nets, Minyi Li, Quoc Bao Vo and Ryszard Kowalczyk, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 353-360.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

(Conditional Preference Networks) [3] is among the most popular ones, since it preserves a good readability that is similar to the way users express their preferences in natural languages. CP-nets provide a compact representation of preference ordering in terms of natural preference statements under a *ceteris paribus* (all else being equal) interpretation. *Ceteris paribus* semantics induces a graph, known as *induced preference graph* [2, 3]; and an outcome α is said to dominate another outcome β if there exists a directed path, also called a *sequence of improving flips*, consisting of successively improving outcomes in the graph from β to α [1, 3]. Unfortunately, reasoning about the preference ordering (dominance relation) expressed by a CP-net is far from easy [3, 5]. With the exception of special cases such as CP-nets with tree or polytree structured conditional dependencies, dominance testing has been shown to be PSPACE-complete even with binary domain and acyclic dependencies [5]. Some general pruning rules have been studied in [3] to reduce the search effort. But they might not be able to guide the search efficiently when the number of variables is large or the structure of the CP-net is complex. Another work proposed by Santhanam *et al.* [9] explores an approach to dominance testing with acyclic CP-nets via Model Checking. However, their approach mainly applies to binary-valued conditional preference statements. The complexity and feasibility of their approach to dominance testing in multi-valued CP-nets is still an open question. Hence, there is a need for efficient algorithms and techniques for dominance testing in more general problem settings.

To this end, we address the problem of dominance testing by proposing an efficient heuristic algorithm, called DT^* , to guide the search process for improving flipping sequence from the worse outcome to the better outcome of the given query¹. The proposed approach can be applied to arbitrary acyclic multi-valued CP-nets. It uses a numerical approximation of the given CP-net and considers the hamming distance between the currently considered outcome and the target outcome of the given query, i.e., the number of variables that the two outcomes differ from each other. We show that our proposed approach efficiently guides the search process for improving flipping sequence. It allows significant reduction of search effort without impacting *soundness* or *completeness* of the search process. Moreover, when there are no flipping sequences possible, it returns the quick failure for the dominance query without having to search all possible branches. We experimentally evaluate the proposed algorithm in different structure settings, including tree-structured CP-nets, directed-path singly connected CP-nets and arbitrary acyclic CP-nets, and with different domain sizes from binary to multi-valued. The experimental results presented in this

¹The proposed heuristic will be described in the context of improving flipping sequences, but it can be applied to worsening search according to the same principle

paper demonstrate that the proposed approach is computationally efficient. It allows dominance queries for CP-nets that are quite large and complex to be answered in reasonable time.

The remainder of this paper is organized as follows. Section 2 restates the necessary background on CP-nets and discusses some existing pruning techniques for the search process in dominance testing. Section 3 introduces the proposed approach in technical details and Section 4 presents the experimental results. Finally, Section 5 discusses the concluding remarks and outlines some directions for future research.

2. PRELIMINARIES

2.1 CP-nets

Let $\mathbf{V} = \{X_1, \dots, X_n\}$ be a set of n variables, for each $X \in \mathbf{V}$, $D(X)$ is the value domain of X . A variable X is binary if $D(X) = \{x, \bar{x}\}$. If $\mathbf{X} = \{X_{i_1}, \dots, X_{i_p}\} \subseteq \mathbf{V}$, with $i_1 < \dots < i_p$ then $D(\mathbf{X})$ denotes $D(X_{i_1}) \times \dots \times D(X_{i_p})$ and \mathbf{x} denotes an assignment of variable values to \mathbf{X} ($\mathbf{x} \in D(\mathbf{X})$). If $\mathbf{X} = \mathbf{V}$, \mathbf{x} is a complete assignment; otherwise \mathbf{x} is called a partial assignment. For any assignment $\mathbf{x} \in D(\mathbf{X})$, we denote by $\mathbf{x}[X]$ the value $x \in D(X)$ ($X \in \mathbf{X}$) assigned to variable X by that assignment; and $\mathbf{x}[\mathbf{W}]$ denotes the assignment of variable values $\mathbf{w} \in D(\mathbf{W})$ assigned to the set of variables $\mathbf{W} \subseteq \mathbf{X}$ by that assignment. If \mathbf{x} and \mathbf{y} are assignments to disjoint sets \mathbf{X} and \mathbf{Y} , respectively ($\mathbf{X} \cap \mathbf{Y} = \emptyset$), we denote the combination of \mathbf{x} and \mathbf{y} by \mathbf{xy} . Let \mathbf{X} , \mathbf{Y} , and \mathbf{Z} be nonempty sets that partition \mathbf{V} and \succ a preference relation over $D(\mathbf{V})$. \mathbf{X} is (conditionally) preferentially independent of \mathbf{Y} given \mathbf{Z} iff for all $\mathbf{x}, \mathbf{x}' \in D(\mathbf{X})$, $\mathbf{y}, \mathbf{y}' \in D(\mathbf{Y})$, $\mathbf{z} \in D(\mathbf{Z})$:

$$\mathbf{xyz} \succ \mathbf{x}'\mathbf{yz} \text{ iff } \mathbf{xy}'\mathbf{z} \succ \mathbf{x}'\mathbf{y}'\mathbf{z}$$

A CP-net \mathcal{N} [3] over \mathbf{V} is an annotated directed graph \mathcal{G} over X_1, \dots, X_n , in which nodes stand for the problem variables. Each node X is annotated with a conditional preference table (CPT), denoted by $CPT(X)$, which associates a total order $\succ^{X|\mathbf{u}}$ with each instantiation \mathbf{u} of X 's parents $Pa(X)$, i.e. $\mathbf{u} \in D(Pa(X))$. For instance, let $\mathbf{V} = \{X_1, X_2, X_3\}$, all three being binary, and assume that the preferences of an agent can be defined by a CP-net whose structural part is the directed acyclic graph $\mathcal{G} = \{(X_1, X_2), (X_1, X_3), (X_2, X_3)\}$; this means that the agent's preference over the values of X_1 is unconditional, preference over the values of X_2 (resp. X_3) is fully determined given the value of X_1 (resp. the values of X_1 and X_2). The preference statements contained in the conditional preference tables are written with the usual notation, that is, $x_1\bar{x}_2 : x_3 \succ \bar{x}_3$ means that when $X_1 = x_1$ and $X_2 = \bar{x}_2$ then $X_3 = x_3$ is preferred to $X_3 = \bar{x}_3$. Figure 1 illustrates an example of CP-net.

2.2 Dominance Testing

One of the most fundamental queries in any preference representation formalism is whether some outcome α dominates (i.e., is strictly preferred to) some other outcome β , called *Dominance Testing*. As discussed in [3, 9], such dominance queries in CP-nets are required whenever we wish to generate more than one non-dominated solutions to a set of hard constraints.

In this paper, we assume the structure of the CP-net is acyclic, i.e. does not contain any dependency cycles. In such case, two outcomes α and β can stand in one of three possible relations with respect to \mathcal{N} : either $\mathcal{N} \models \alpha \succ \beta$ (α is strictly preferred to β); or $\mathcal{N} \models \beta \succ \alpha$ (β is strictly preferred to α); or $\mathcal{N} \models \alpha \bowtie \beta$ (α and β are incomparable: $\mathcal{N} \not\models \alpha \succ \beta$ and $\mathcal{N} \not\models \beta \succ \alpha$). The third case means that the given CP-net \mathcal{N} does not contain enough

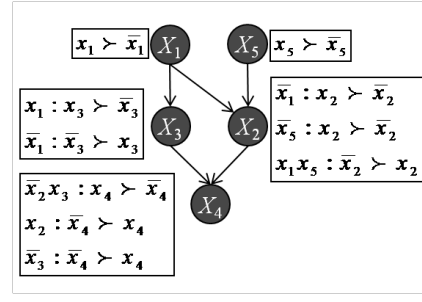


Figure 1: An example CP-net \mathcal{N}

information to prove that either outcome is preferred to the other. Given an acyclic CP-net, comparisons between two outcomes that differ in the value of a single variable are easy: we only need to check the CPT of that variable and determine which outcome assigns it to a more preferred value. The better (improved) outcome can be considered as a product of a single *improving flip* in the value of a variable X from the worse outcome. For any pair of outcomes that differ on more than one variable, an outcome α is said to dominate another outcome β with respect to an acyclic CP-net \mathcal{N} ($\mathcal{N} \models \alpha \succ \beta$) if there exists a *sequence of improving flips* from β to α . Otherwise, $\mathcal{N} \not\models \alpha \succ \beta$. The following definition of improving flipping sequence is introduced in [3].

DEFINITION 1 (IMPROVING FLIPPING SEQUENCE).

A sequence of outcomes $\beta = \gamma_1, \gamma_2, \dots, \gamma_{m-1}, \gamma_m = \alpha$ such that

$$\beta = \gamma_1 \prec \gamma_2 \prec \dots \prec \gamma_{m-1} \prec \gamma_m = \alpha$$

is an improving flipping sequence with respect to an acyclic CP-net \mathcal{N} if and only if, $\forall 1 \leq i \leq m$, outcome γ_i is different from the outcome γ_{i+1} in the value of exactly one variable X , and $\gamma_{i+1}[X] \succ \gamma_i[X]$ given the parent context \mathbf{u} of X assigned by γ_i and γ_{i+1} .

For instance, consider the preference statements over two binary variables X_1 and X_2 , $x_1 \succ \bar{x}_1$, $x_2 \succ \bar{x}_2$, the sequence $\bar{x}_1\bar{x}_2, \bar{x}_1x_2, x_1x_2$ is an improving flipping sequence from the outcome $\bar{x}_1\bar{x}_2$ to the best outcome x_1x_2 .

2.3 Some General Search Techniques

Given an acyclic CP-net \mathcal{N} , a query $\mathcal{N} \models \alpha \succ \beta$ can be treated as a search for an improving flipping sequence from the less preferred outcome β to the more preferred outcome α . The search process can be implemented as an *improving search tree* rooted at β , $T(\beta)$. The children of every node² γ in $T(\beta)$ are those outcomes that can be reached by a single improving flip from γ . Consequently, every rooted path in $T(\beta)$ corresponds to some improving flipping sequence from the outcome β with respect to \mathcal{N} . Taking different directions in $T(\beta)$ leads to different improving sequences; however, taking a different direction during the tree traversal may also lead to a dead end, i.e., reach the optimal outcome of \mathcal{N} without visiting the target outcome α of the query. Recent works have studied the computational complexity of testing dominance relations in CP-nets, e.g. [3, 5]. The results show that dominance testing in general CP-nets is PSPACE-complete and it remains PSPACE-complete even though the CP-net is acyclic [5]. Since the hardness of dominance testing, several search techniques for dominance queries have been studied in [3] in order to reduce the search effort.

Suffix Fixing. Let $X_{i_1} > \dots > X_{i_n}$ be an arbitrary topological ordering consistent with the CP-net \mathcal{N} , an r th ($r \geq 1$)

²A node in the improving search tree is also an outcome.

suffix of an outcome α is the subset of the outcome values $\alpha[X_{i_r}] \alpha[X_{i_{r+1}}] \dots \alpha[X_{i_n}]$. The r th suffix of outcomes α and β match iff $\forall r \leq j \leq n, \alpha[X_{i_j}] = \beta[X_{i_j}]$. For a query $\mathcal{N} \models \alpha \succ \beta$, suffix fixing rules out the exploration of any possible flipping sequences that destroy of the suffix of the currently considered outcome that matches the target outcome α . It prunes the subtree that improves the value of a variable within the matching suffix. For instance, consider the CP-net \mathcal{N} in Figure 1 and the query $\mathcal{N} \models x_1 \bar{x}_2 x_3 x_4 x_5 \succ \bar{x}_1 x_2 x_3 x_4 \bar{x}_5$. Let $\alpha = x_1 \bar{x}_2 x_3 x_4 x_5$ and $\beta = \bar{x}_1 x_2 x_3 x_4 \bar{x}_5$, if pruned using suffix fixing and consider the variable ordering $X_1 > X_5 > X_2 > X_3 > X_4$, the 2nd suffix $x_3 x_4$ of α and β matches. Thus, the values of X_3 and X_4 will never be improved in the search tree $T(\beta)$, although given the assignment $\beta[X_1] = \bar{x}_1$ (resp. $\beta[X_2] = x_2, \beta[X_3] = x_3, \bar{x}_3 \succ x_3$ (resp. $\bar{x}_4 \succ x_4$). As shown in [3], any complete search algorithm for the improving search tree remains complete if pruning using suffix fixing is used.

Least-variable flipping. For every node γ in the improving search tree, least-variable flipping rule restricts flips to the variables that are least-improvable. Formally, a variable X is least-improvable in an outcome γ with respect to \mathcal{N} if there is some value $x \in D(X)$ such that $x \succ_{\mathbf{u}} \gamma[X]$ (where $\mathbf{u} = \gamma[Pa(X)]$ is the parent context assigned by γ), and no descendant of X in γ has this property. For a query $\mathcal{N} \models \alpha \succ \beta$, *least-variable flipping rule* restricts attention to those variables that are not part of any matching suffix with the target outcome α and requires that the only neighbours of a node γ can be expanded in the search tree $T(\beta)$ are those in which some least improvable variable with respect to γ is improved.

However, least-variable flipping rule is only complete for a restricted class of CP-nets [3], i.e. tree-structured CP-nets and binary-valued, directed-path singly connected CP-nets. For multiply-connected networks, and networks with multi-valued variables, it does not guarantee completeness. That means, least-variable flipping may fail to find any improving sequence from β to α although there does exist at least one. In such case, it does not provide a correct answer to the given query. For instance, consider the CP-net \mathcal{N} in Figure 1 and the query $\mathcal{N} \models x_1 x_2 x_3 x_4 x_5 \succ \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$.³ Starting with the root node $\beta = \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$, the only least improvable variable that can be flipped is X_2 . Unfortunately, flipping X_2 to value x_2 leads to outcome $\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$, from which the target outcome $\alpha = x_1 x_2 x_3 x_4 x_5$ is unreachable. All branches in the improving search tree grow towards the optimal outcome $x_1 \bar{x}_2 x_3 x_4 x_5$ without going through the target outcome α of the query. Figure 2 shows the complete improving search tree $T(\beta)$ using least-variable flipping. However, there in fact exists a sequence of improving flips from β to α : $\bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5, x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5, x_1 \bar{x}_2 x_3 \bar{x}_4 \bar{x}_5, x_1 \bar{x}_2 x_3 x_4 \bar{x}_5, x_1 x_2 x_3 x_4 \bar{x}_5, x_1 x_2 x_3 x_4 x_5$.

When the number of variables is large or the structure of the CP-net is complex, suffix fixing may not be able to guide the search efficiently while least-variable flipping rule does not guarantee completeness for general acyclic CP-nets. To this end, we will present another efficient heuristic approach to dominance testing. The proposed approach significantly prunes the search tree without impacting soundness or completeness of the search process.

³This example has also been discussed in Example 7 in [3]

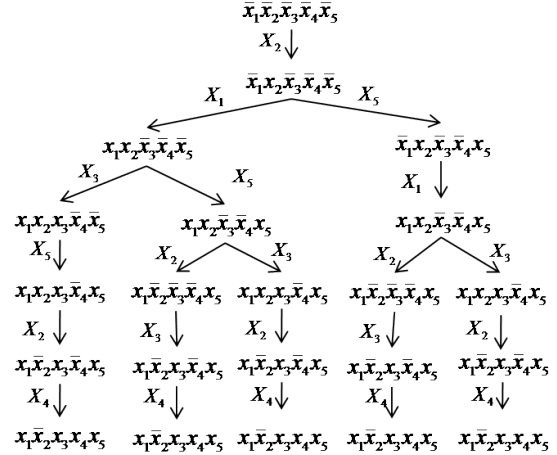


Figure 2: Improving search tree for query $\mathcal{N} \models x_1 x_2 x_3 x_4 x_5 \succ \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$ using Least-variable flipping rule

3. HEURISTIC FOR DOMINANCE TESTING

In this section, we present our proposed heuristic approach, called DT^* , to dominance testing in arbitrary acyclic CP-nets. In broad terms, we first define a penalty function based on a numerical approximation proposed by Domshlak *et al.* [4] that approximates acyclic CP-nets using weighted soft constraints. Then, an evaluation function is defined based on the hamming distance between the currently considered outcome and the target outcome and their penalties as a heuristic to guide the search process.

3.1 Penalty function

For a variable X , let $|D(X)|$ be the domain size of X and thus there are $|D(X)|$ degrees of penalties of X , denoted by $d_1, \dots, d_{|D(X)|}$. Without loss of generality, we assume the degree of penalties of a variable X range between 0 and $|D(X)| - 1$; that is, $d_1 = 0, \dots, d_{|D(X)|} = |D(X)| - 1$. For instance, consider the agent's CP-net in Figure 1, since all variables are binary, there are only two degrees of penalties, i.e., $d_1 = 0$ and $d_2 = 1$ for each variable. For a variable X , consider a preference ordering over the value of X given an instantiation of X 's parents, let the rank of the most preferred value of X be 0 and the rank of the least preferred value of X be $|D(X)| - 1$, given an outcome γ , the degree of penalty of a variable X in γ is then the rank of the value $\gamma[X]$ in the preference ordering over X given the parent context $\mathbf{u} = \gamma[Pa(X)]$. We denote by d_X^r ($d_X^r \in \{d_1, \dots, d_{|D(X)|}\}$) the degree of penalty of X with respect to γ . For instance, consider a variable X such that $D(X) = \{x, x', x''\}$. Assume that, under a parent context $\mathbf{u} = \gamma[Pa(X)]$ assigned by an outcome γ , $x \succ x' \succ x''$. If $\gamma[X] = x$, then $d_X^r = d_1 = 0$; if $\gamma[X] = x'$, then $d_X^r = d_2 = 1$; if $\gamma[X] = x''$, then $d_X^r = d_3 = 2$.

CP-net imposes a rich structure to allow variables to have different degrees of importance: variables "higher-up" in the structure of the network are considered to be more important than the lower level variables [1, 2, 3]. Thus, it is more important to obtain a preferred value for a variable than any of its descendants. We now analyse the *importance weight* of a variable in a CP-net. Given an acyclic CP-net \mathcal{N} and consider an improving flip from an outcome γ to another outcome γ' that flips the value of a single variable X , changing the value of X may also affect the preference status of X 's children. Thus, the resulting changes from γ to γ' includes: (i) the degree of penalty of X decreases from d_X^r to $d_X^{r'}$ ($d_X^r > d_X^{r'}$);

Algorithm 1: `assgWeightCP(\mathcal{N})`

Input: \mathcal{N} , an acyclic CP-net
1 Order variables of \mathcal{N} in a reverse topological ordering;
2 **foreach** $X \in \mathcal{N}$ **do**
3 **if** $Ch(X) = \emptyset$ **then**
4 $w_X \leftarrow 1$;
5 **else**
6 $w_X \leftarrow 1 + \sum_{Y \in Ch(X)} w_Y \cdot (|D(Y)| - 1)$;
7 **end**
8 **end**

and (ii) the degrees of penalty of X 's children changes, which in the worst case, results in the degree of penalty of each child Y increasing from $d_Y^\gamma = d_1$ to $d_Y^{\gamma'} = d_{|D(Y)-1|}$. Consequently, in order to preserve the preference ordering induced by the given CP-net, the importance weight of a variable in that CP-net must be larger than the sum of the maximum penalties of its children. We now provide the formal definition of the variable importance weight in an acyclic CP-net.

DEFINITION 2 (IMPORTANCE WEIGHT). *Given an acyclic CP-net \mathcal{N} over a set of variables \mathbf{V} . For each variable $X \in \mathbf{V}$, let $Ch(X)$ denote the set of children of X in \mathcal{N} , the importance weight of variable X , denoted by w_X , is recursively defined by:*

$$w_X = 1 + \sum_{Y \in Ch(X)} w_Y \cdot (|D(Y)| - 1) \quad (1)$$

Algorithm 1 provides a simple implementation to compute importance weights of variables. It takes *linear* time in the size of the network. Following a reverse topological ordering, it first assigns the importance weights to the variables that have no descendants (line 3–4) and then iteratively assigns the importance weights to the upper level variables according to Equation (1). Note that there are several ways to assign importance weights to the variables and the way we use here is different from [4]. In this paper, we consider the tight lower bound of the importance weight assignment, i.e. the sum of maximum penalties of the variable children $\sum_{Y \in Ch(X)} w_Y \cdot (|D(Y)| - 1)$.

EXAMPLE. Consider an agent's CP-net over a set of 5 variables $\mathbf{V} = \{X_1, \dots, X_5\}$ in Figure 1. In this example, since all variables are binary, i.e. $\forall X \in \mathbf{V}, |D(X)| = 2$. We can assign the importance weight to each variable in a reverse topological ordering of variables: $w_{X_4} = 1$; $w_{X_2} = 1 + w_{X_4} \cdot (2 - 1) = 2$; $w_{X_3} = 1 + w_{X_4} \cdot (2 - 1) = 2$; $w_{X_1} = 1 + (w_{X_2} \cdot (2 - 1) + w_{X_3} \cdot (2 - 1)) = 5$; $w_{X_5} = 1 + w_{X_2} \cdot (2 - 1) = 3$. The importance weight of each variable in this CP-net is attached on top of the variables respectively in Figure 3.

Given an acyclic CP-net \mathcal{N} and an outcome γ , the *penalty* of a variable X in γ is the *degree of penalty* of X in γ multiplied by the *importance weight* of X . The penalty of γ is then defined by the sum of penalties of the domain variables. We define the following penalty function for an acyclic CP-net based on the work by Domshlak *et al.* [4].

DEFINITION 3 (PENALTY FUNCTION). *Given an acyclic CP-net \mathcal{N} over a set of variables \mathbf{V} and an outcome γ . The penalty function pen , mapping from an outcome $\gamma \in O$ to $[0, +\infty]$, is defined as follows:*

$$\forall \gamma \in O, pen(\gamma) = \sum_{X \in \mathbf{V}} w_X \cdot d_X^\gamma \quad (2)$$

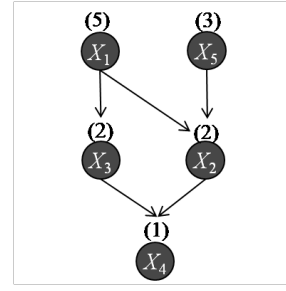


Figure 3: Variable importance weight in \mathcal{N}

EXAMPLE (CONT.) Consider our running example in Figure 1 and the outcome $\gamma = \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$. As the agent unconditionally prefers $X_1 = x_1$ to $X_1 = \bar{x}_1$ (resp. $X_5 = x_5$ to $X_5 = \bar{x}_5$), $d_{X_1}^\gamma = 1$ (resp. $d_{X_5}^\gamma = 1$). On the other hand, $x_2 \succ \bar{x}_2$ (resp. $\bar{x}_3 \succ x_3$, $\bar{x}_4 \succ x_4$) given the parent context $X_1 = \bar{x}_1$ and $X_5 = \bar{x}_5$ (resp. $X_1 = \bar{x}_1$, $X_2 = \bar{x}_2$ and $X_3 = \bar{x}_3$) and thus $d_{X_2}^\gamma = 1$ (resp. $d_{X_3}^\gamma = 0$, $d_{X_4}^\gamma = 0$). Consequently, the penalty of outcome $\bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$ is: $pen(\gamma) = w_{X_1} \cdot 1 + w_{X_2} \cdot 1 + w_{X_3} \cdot 0 + w_{X_4} \cdot 0 + w_{X_5} \cdot 1 = 5 \cdot 1 + 2 \cdot 1 + 2 \cdot 0 + 1 \cdot 0 + 3 \cdot 1 = 10$.

In order to compute the penalty of an outcome, we simply need to sweep through the network from top to bottom (i.e., from ancestors to descendants), and to check the degree of penalty of the currently considered variable given its parent context. And finally we compute the penalty of the outcome based on Equation (2). Consequently, the penalty computation for a particular outcome takes *polynomial time* in the size of the network. We now prove that our algorithm for assigning penalties over alternative outcomes preserves the strict preference ordering induced by the original CP-net.

THEOREM 1. *Given an acyclic CP-net \mathcal{N} , we have:*

$$\forall \alpha, \beta \in O, \text{ if } \mathcal{N} \models \alpha \succ \beta \text{ then } pen(\beta) > pen(\alpha)$$

PROOF. $\mathcal{N} \models \alpha \succ \beta$ if and only if there exists a sequence of improving flips from β to α , denoted by $Seq(\beta, \alpha) = \gamma_1 (= \beta), \gamma_2, \dots, \gamma_{m-1}, \gamma_m (= \alpha)$, with respect to the conditional preference tables in \mathcal{N} . Each improving flip from γ_i to γ_{i+1} in $Seq(\beta, \alpha)$ that improves the value of a single variable X , $pen(\gamma_i) - pen(\gamma_{i+1}) = w_X \cdot (d_X^{\gamma_i} - d_X^{\gamma_{i+1}}) + \sigma$, where $\sigma \geq -\sum_{Y \in Ch(X)} w(Y) \cdot (|D(Y)| - 1)$ and $(d_X^{\gamma_i} - d_X^{\gamma_{i+1}}) \geq 1$. Thus, $pen(\gamma_i) - pen(\gamma_{i+1}) \geq w_X - \sum_{Y \in Ch(X)} w(Y) \cdot (|D(Y)| - 1) = w_X - (w_X - 1) = 1 > 0$. Consequently, with each improving flip from γ_i to γ_{i+1} , $pen(\gamma_i) > pen(\gamma_{i+1})$. Following from the transitivity: $pen(\gamma_1 (= \beta)) > pen(\gamma_2) > \dots > pen(\gamma_{m-1}) > pen(\gamma_m (= \alpha))$ and thus $pen(\beta) > pen(\alpha)$. \square

COROLLARY 1. *Given an acyclic CP-net \mathcal{N} , $\forall \alpha, \beta \in O$,*

- *if $pen(\beta) > pen(\alpha)$ then $\mathcal{N} \models \alpha \succ \beta$ or $\mathcal{N} \models \alpha \bowtie \beta$*
- *if $pen(\beta) = pen(\alpha)$ then $\mathcal{N} \models \alpha \bowtie \beta$*

LEMMA 1. *Given an acyclic CP-net \mathcal{N} over a set of variables \mathbf{V} , let α, β be any pair of outcomes that $\mathcal{N} \models \alpha \succ \beta$; **IS** the set of all possible improving flipping sequence from β to α with respect to the CPTs in \mathcal{N} ; $HD(\beta, \alpha)$ the hamming distance between β*

and α (Note that both in binary-valued and multi-valued CP-nets, the hamming distance is defined by the number of variables that the two outcomes differ from each other.); $Seq(\beta, \alpha) \in \mathbf{IS}$ an improving flipping sequence from β to α ; $|Seq(\beta, \alpha)|$ is the length of $Seq(\beta, \alpha)$ and thus the number of improving flips from β in this sequence is $|Seq(\beta, \alpha)| - 1$, then,

$$HD(\beta, \alpha) \leq |Seq(\beta, \alpha)| - 1 \leq pen(\beta) - pen(\alpha)$$

PROOF. As each improving flip flips the value of a single variable, if $\mathcal{N} \models \alpha \succ \beta$, there must be at least $HD(\beta, \alpha)$ flips that flips the value of each variable X that β and α differ, from $\beta[X]$ to $\alpha[X]$. Thus, $\forall Seq(\beta, \alpha) \in \mathbf{IS}$, $|Seq(\beta, \alpha)| - 1 \geq HD(\beta, \alpha)$. On the other hand, any improving flip from γ_i to γ_{i+1} in $Seq(\beta, \alpha)$ that flips the value of a single variable X , $pen(\gamma_i) - pen(\gamma_{i+1}) \geq 1$ (see the proof of Theorem 1). Thus, $pen(\gamma_i) - pen(\gamma_{i+1}) \geq 1$. Assume γ is an outcome in $Seq(\beta, \alpha)$ that improved from β by t flips, $pen(\beta) - pen(\gamma) \geq t$ and $pen(\beta) - t \geq pen(\gamma)$. Thus, $pen(\beta) - pen(\alpha) - t \geq pen(\gamma) - pen(\alpha)$. If $t > pen(\beta) - pen(\alpha)$, then $pen(\beta) - pen(\alpha) - t < 0$ and $pen(\gamma) - pen(\alpha) < 0$. According to Corollary 1, $\mathcal{N} \models \gamma \succ \alpha$ or $\mathcal{N} \models \gamma \bowtie \alpha$ ($\mathcal{N} \not\models \alpha \succ \gamma$), contradicting the fact that γ is in the improving sequence $Seq(\beta, \alpha)$. Hence, the number of improving flips from β in $Seq(\beta, \alpha)$ can not be greater than $pen(\beta) - pen(\alpha)$, $|Seq(\beta, \alpha)| - 1 \leq pen(\beta) - pen(\alpha)$. \square

3.2 The proposed DT* algorithm

The penalty function mentioned above provides an order-preserving numerical approximation for a given CP-net. We also show the upper bound and lower bound of the number of improving flips from a worse outcome to a preferred outcome in Lemma 1. In this section, these results are used as a heuristic in the search process for improving flipping sequence. The proposed algorithm has a number of desirable properties:

- it often returns the quick failure for the dominance query if no flipping sequence is possible;
- it often quickly shows that back-tracking is needed when there is no possible flipping sequence to the target outcome following the currently considered path; and,
- it efficiently guides the search direction without compromising soundness or completeness of the search process.

Given an acyclic CP-net \mathcal{N} and a pair of outcomes α and β , for the query $\mathcal{N} \models \alpha \succ \beta$, we build the search tree $T(\beta)$ and search for an improving flipping sequence to the target outcome α as discussed in [3]. We introduce the evaluation function f for the heuristic search strategy as follows:

DEFINITION 4 (EVALUATION FUNCTION). *Given an acyclic CP-net \mathcal{N} and the query $\mathcal{N} \models \alpha \succ \beta$ ($\alpha, \beta \in O$). The evaluation function f , mapping from a node (i.e., an outcome) γ in the improving search tree $T(\beta)$ to $[0, +\infty]$, is defined by:*

$$f(\gamma) = pen(\gamma) - HD(\gamma, \alpha) - pen(\alpha) \quad (3)$$

Our proposed heuristic algorithm DT* (see Algorithm 2) is adapted from the A* heuristic search algorithm with $f(\gamma)$ being the evaluation function. It maintains a priority queue of nodes to be expanded, known as the *fringe*. On the one hand, the lower f value for a node γ , the higher its priority is. On the other hand, we only consider the outcomes that the f value is non-negative. That means,

Algorithm 2: DT*($\mathcal{N} \models \alpha \succ \beta$)

Input: a dominance query (an acyclic CP-net \mathcal{N} ; a pair of outcomes α and β ; and determining whether $\mathcal{N} \models \alpha \succ \beta$)
Output: *True:* $\mathcal{N} \models \alpha \succ \beta$; *False:* $\mathcal{N} \not\models \alpha \succ \beta$

```

1 if  $f(\beta) < 0$  then
2   return False;
3 else
4    $fringe \leftarrow \text{INSERT}(\text{MAKE-NODE}(\beta), fringe)$ ;
5   while  $fringe \neq \emptyset$  do
6      $\gamma^* \leftarrow \text{REMOVE-FIRST}(fringe)$ ;
7     if  $\text{GOAL-TEST}(\gamma^* = \alpha)$  then
8       return True;
9     else
10      foreach  $X \in \mathbf{N}$  do
11        if  $\text{IMPROVABLE}(\gamma^*, X)$ 
12          &&  $X \notin \text{ANY-MATCHING-SUFFIX}(\gamma^*, \alpha)$ 
13          then
14             $\gamma' \leftarrow \text{SINGLE-FLIP}(\gamma^*, X)$ ;
15            if  $\text{NOT-REPEATED}(\gamma')$  &&  $f(\gamma') \geq 0$ 
16              then
17                 $\text{INSERT-ASC}(\text{MAKE-NODE}(\gamma'), fringe)$ 
18              end
19            end
20          end
21   return False

```

an outcome γ will be added into the *fringe* only if $f(\gamma) > 0$. In essence, an outcome with a negative f value means that there is no possible improving flipping sequence from that outcome to the target outcome α (see Lemma 2). Before adding the original node β into the *fringe*, the f value of β will be computed and the algorithm will return *False* if $f(\beta) < 0$ (line 1–2). In this case, the query fails ($\mathcal{N} \not\models \alpha \succ \beta$) even before building the root node of the improving search tree. Otherwise, β will be added into the *fringe* as the root node of the improving search tree $T(\beta)$ (line 4). At each iteration of DT*, the first node γ^* , i.e. the node with the lowest f value, is removed from the *fringe* and being expanded (line 4). The children of a node in $T(\beta)$ are those outcomes that can be reached by a single improving flip from that node. Our proposed algorithm applies suffix fixing rules, restricting attention to those variables in γ^* that are not part of any matching suffix with the target outcome α (line 11). Moreover, it requires that a child γ' of a node be added into the *fringe* if and only if: (i) γ' has not been traversed before; and (ii) $f(\gamma') \geq 0$ (line 13). For the current node γ^* under consideration, we add each child γ' of γ^* that meets the above requirements into the *fringe* in ascendant order of the f values of the nodes in the *fringe* (line 14). DT* continues until: the currently considered node for expansion equals to the target outcome α , then it ends and returns *True* ($\mathcal{N} \models \alpha \succ \beta$) (line 7–8); or the *fringe* is empty, it returns *False* ($\mathcal{N} \not\models \alpha \succ \beta$) (line 20).

In order to prove the completeness of our proposed heuristic algorithm, we first proof the follow lemma.

LEMMA 2. *Given an acyclic CP-net \mathcal{N} and a query $\mathcal{N} \models \alpha \succ \beta$ ($\alpha, \beta \in O$), $\forall \gamma^* \in O$, if $f(\gamma^*) < 0$, then γ^* would not be part of any possible improving flipping sequence from β to α .*

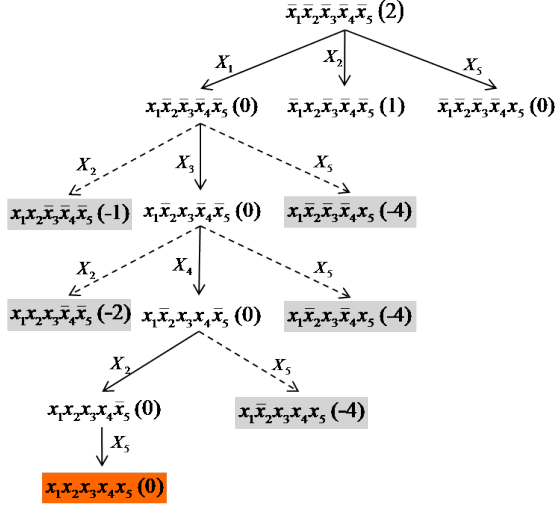


Figure 4: Improving search tree

PROOF. During the execution of DT^* algorithm, for any outcome γ^* (including β), $f(\gamma^*) = pen(\gamma^*) - HD(\gamma^*, \alpha) - pen(\alpha)$. Assume that there exist an improving flipping sequence $Seq(\gamma^*, \alpha) = \gamma_1 (= \gamma^*), \gamma_2, \dots, \gamma_{m-1}, \gamma_m (= \alpha)$ from γ^* to the target outcome α . Based on Lemma 1, we know that there must be at least $HD(\gamma^*, \alpha)$ flips improved from γ^* . For any improving flip from γ_i to γ_{i+1} , $pen(\gamma_i) - pen(\gamma_{i+1}) \geq 1$. Consequently, for any outcome γ' that improved from γ^* by $HD(\gamma^*, \alpha)$ flips, $pen(\gamma^*) - pen(\gamma') \geq HD(\gamma^*, \alpha)$ and thus $pen(\gamma^*) - HD(\gamma^*, \alpha) \geq pen(\gamma')$. Hence, $pen(\gamma^*) - HD(\gamma^*, \alpha) - pen(\alpha) \geq pen(\gamma') - pen(\alpha)$. Because $f(\gamma^*) < 0$, $pen(\gamma') - pen(\alpha) \leq pen(\gamma^*) - HD(\gamma^*, \alpha) - pen(\alpha) < 0$. Consequently, $pen(\gamma') - pen(\alpha) < 0$ and $\mathcal{N} \not\models \alpha \succ \gamma'$. γ' will not be part of any possible improving flipping sequence to α , contradicting the fact that there exist an improving flipping sequence $Seq(\gamma^*, \alpha)$ from γ^* to the target outcome α . \square

We now prove the completeness of our proposed heuristic algorithm.

THEOREM 2. DT^* is complete for any arbitrary acyclic CP-nets.

PROOF. DT^* traverses the tree searching all neighbours; it follows lowest evaluated value path and keeps a sorted priority queue of alternate path segments along the way. If at any point the path being followed has a higher evaluated value than other encountered path segments, the higher evaluated value path is kept in the *fringe* and the process is continued at the lower value sub-path. This continues until the currently considered node for expansion is the target outcome or the fringe is empty. During the execution of DT^* algorithm, there are three kinds of nodes will be pruned: (i) the outcomes that have been explored previously; (ii) the outcomes that improve the value of the variable that is part of some matching suffix with the target outcome; and (iii) the outcomes with negative f values. Obviously, checking repeated nodes does not affect the completeness of the algorithm. Also, as shown in [3], any complete search algorithm for the improving search tree remains complete if pruning using suffix fixing rule is used. Furthermore, we have already proved in Lemma 2 that an outcome γ^* with $f(\gamma^*) < 0$ will not be part of any possible improving sequence from β to α , so pruning the third kind of outcomes also does not affect the completeness of the algorithm. Consequently, DT^* is complete for any acyclic CP-nets. \square

EXAMPLE (CONT.) We now demonstrate the execution of DT^* algorithm with the CP-net in our running example (Figure 1) and consider the query $\mathcal{N} \models x_1 x_2 x_3 x_4 x_5 \succ \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$. Let $\alpha = x_1 x_2 x_3 x_4 x_5$ and $\beta = \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$, we first consider the f value of the less preferred outcome β of the query. As $f(\beta) = pen(\beta) - HD(\beta, \alpha) - pen(\alpha) = 10 - 5 - 3 = 2 > 0$, we build the search tree $T(\beta)$ with β being the root node and add β into the *fringe*. In the 1th iteration of DT^* , $\gamma^* = \bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$ is removed from the *fringe* to be expanded. There are three improvable variable from γ^* : X_1 , X_2 and X_5 . Hence, there are three children nodes: $x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$, $\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$ and $\bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 x_5$. The f value of these three children nodes are computed accordingly. $f(x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5) = 0$, $f(\bar{x}_1 x_2 \bar{x}_3 \bar{x}_4 \bar{x}_5) = 1$ and $f(\bar{x}_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 x_5) = 0$. As none of the f value of these three children nodes is negative, all of them are added into the *fringe* according to the ascendant order of the f value.

In the 2nd iteration, the first outcome $\gamma^* = x_1 \bar{x}_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$ with the lowest f value is removed from the *fringe* (Assume that the nodes with the same f value will be traversed in the order from left to right). There are three possible children nodes of γ^* : $x_1 x_2 \bar{x}_3 \bar{x}_4 \bar{x}_5$, $x_1 \bar{x}_2 x_3 \bar{x}_4 \bar{x}_5$ and $x_1 \bar{x}_2 \bar{x}_3 x_4 \bar{x}_5$. As $f(x_1 x_2 \bar{x}_3 \bar{x}_4 \bar{x}_5) = 5 - 3 - 3 = -1 < 0$; $f(x_1 \bar{x}_2 x_3 \bar{x}_4 \bar{x}_5) = 6 - 3 - 3 = 0$; and $f(x_1 \bar{x}_2 \bar{x}_3 x_4 \bar{x}_5) = 2 - 3 - 3 = -4 < 0$. There is only one outcome $x_1 \bar{x}_2 x_3 \bar{x}_4 \bar{x}_5$ will be added into the *fringe*.

In the 3rd iteration, we continue with the outcome $\gamma^* = x_1 \bar{x}_2 x_3 \bar{x}_4 \bar{x}_5$. There are three possible outcomes can be reached by a single flip from γ^* : $x_1 x_2 x_3 \bar{x}_4 \bar{x}_5$, $x_1 \bar{x}_2 x_3 x_4 \bar{x}_5$ and $x_1 \bar{x}_2 x_3 \bar{x}_4 x_5$. We compute the f value of these three outcomes: $f(x_1 x_2 x_3 \bar{x}_4 \bar{x}_5) = 3 - 2 - 3 = -2 < 0$; $f(x_1 \bar{x}_2 x_3 x_4 \bar{x}_5) = 5 - 2 - 3 = 0$; and $f(x_1 \bar{x}_2 x_3 \bar{x}_4 x_5) = 1 - 2 - 3 = -4 < 0$. Only one outcome $x_1 \bar{x}_2 x_3 x_4 \bar{x}_5$ can be added into the *fringe*.

Similarly, in the 4th iteration, we explore the outcome $\gamma^* = x_1 \bar{x}_2 x_3 x_4 \bar{x}_5$ and add only one outcome $x_1 x_2 x_3 x_4 \bar{x}_5$ into the *fringe*.

In the 5th iteration, we explore the outcome $\gamma^* = x_1 x_2 x_3 x_4 \bar{x}_5$. In essence, there are two variables can be improved from γ^* : X_4 and X_5 . However, as X_4 is in the 3rd matching suffix with the target outcome α (using the topological order $X_1 > X_5 > X_2 > X_3 > X_4$), we only consider flipping the value of X_5 . And this step produces the target outcome α , which will be explored in the last iteration and the algorithm returns *True* to this query.

Note that as we have discussed in Section 2.3, an algorithm based on Least-variable flipping rule is incomplete in this case.

4. EXPERIMENT

We now describe the results of experiments that show the feasibility of our approach to dominance testing with respect to (i) the average number of visited nodes during the search process; (ii) the number of variables s_{var} and the domain size s_{dz} that can be efficiently handled in practice; and (iii) the structure of CP-nets. We compare the performance of the proposed DT^* algorithm with (i) a standard depth-first search algorithm that applies suffix fixing during the search, called DF; and (ii) an algorithm using Least-variable flipping rule, called LVF. We generate random preference networks by varying the number of variables, the structure of the network and the preference of the variables. For directed-path singly connected CP-nets and arbitrary acyclic CP-nets, we restrict the maximum in-degree of each node in the generated CP-nets to 10. For multi-valued CP-nets, we restrict the maximum domain size s_{dz} to 5. We conduct the following six sets of experiments. At each set of experiments, we generate 1000 CP-nets randomly and using each resulting preference network, we evaluate 5 dominance queries by picking distinct pairs of outcomes at random.

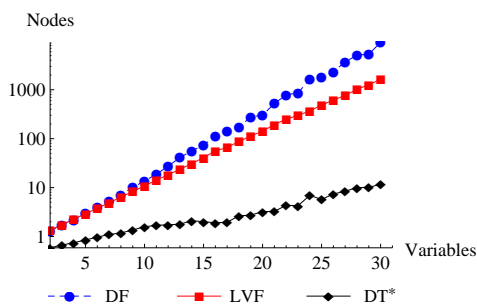


Figure 5: Avg. number of visited nodes with binary-value tree-structured CP-nets

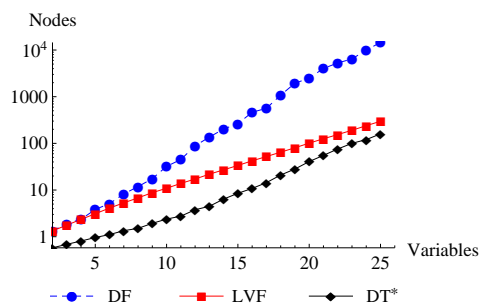


Figure 7: Avg. number of visited nodes with binary-valued, directed-path singly connected CP-nets

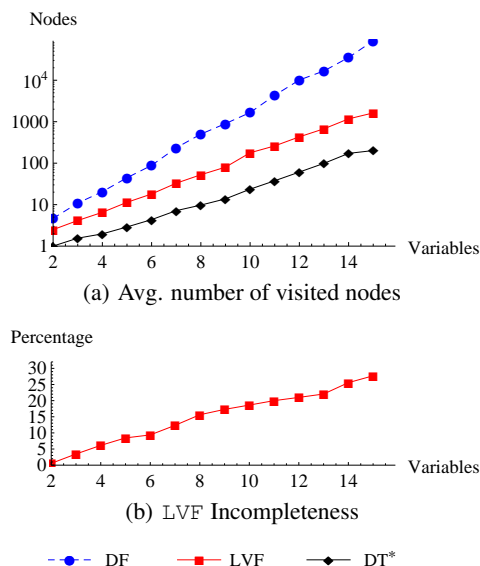


Figure 6: Multi-valued tree-structured CP-nets

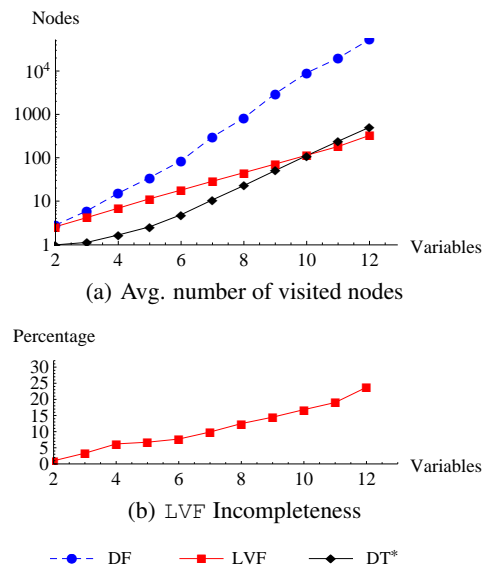


Figure 8: Multi-valued polytree CP-nets

Set 1: binary-valued tree-structured CP-nets. We vary the number of variables s_{var} from 2 to 30 and only generate tree-structured dependences. From Figure 5 we can observe that on average, the numbers of visited nodes by both DT^* and LVF algorithms are much less than DF algorithm. Note that for binary-valued tree-structured CP-nets, LVF (Least-variable flipping rule) is guaranteed to be *complete* and *backtrack-free*.⁴ However, on average, DT^* is more efficient than the LVF algorithm for dominance testing in tree-structured CP-nets. The average execution time of DT^* approach with 30 variables is less than 0.03 seconds. It offers more than three orders of magnitude improvement in performance over the DF algorithm.

Set 2: multi-valued tree-structured CP-nets. We vary s_{var} from 2 to 15. The results of multi-valued tree-structured CP-nets (see Figure 6(a)) is similar to the set of experiments with binary-valued tree-structured CP-nets. However, LVF algorithm does not guarantee completeness in multi-valued CP-nets. Figure 6(b) shows the percentage of cases in which the LVF algorithm is incomplete, i.e., it gives an incorrect

answer to the query. In general, the percentage of incompleteness of LVF algorithm is increasing as the number of variables increases. When there are 15 variables, in more than 28% cases that LVF algorithm fails to find the improving flipping sequence for the given query although there does exist at least one. On the other hand, according to the experiment data, DT^* completes the search process in about 12 seconds on average in the cases of 15 variables.

Set 3: binary-valued, directed-path singly connected CP-nets. In this set of experiments, the number of variables s_{var} is from 2 to 25. Note that LVF algorithm guarantees completeness in binary-valued, directed-path singly connected CP-nets while it may require back-tracking during the search. The average number of visited nodes in this set of experiments is shown in Figure 7. Both LVF and DT^* algorithms are much more efficient than the DF algorithm. When there are 25 variables, the average execution time of DT^* is about 5.7 seconds, which is more than two orders of magnitude less than the DF algorithm.

Set 4: multi-valued, directed-path singly connected CP-nets. We vary s_{var} from 2 to 12. Figure 8(a) shows that the average number of visited nodes of both LVF and DT^* algorithms are much less than DF algorithm. Although the result shows that

⁴The authors can also refer to [3] Page 161, *TreeDT* algorithm for binary-valued, tree-structured CP-nets

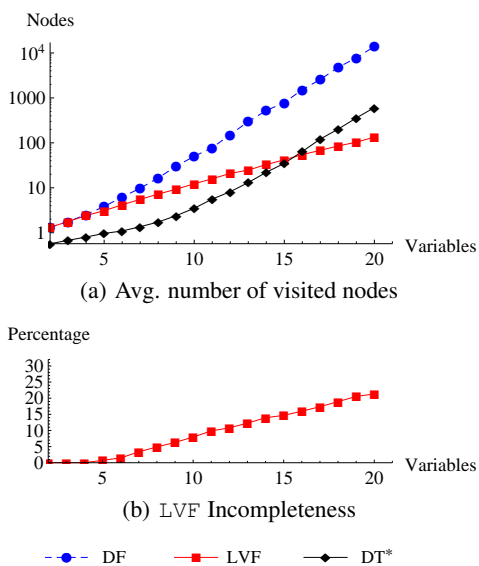


Figure 9: Binary-value arbitrary acyclic CP-nets

when the number of variables is large, the LVF algorithm may visit less nodes than DT* algorithm, the percentage of incompleteness of LVF is on the other hand, increasing as the number of variables increases (see Figure 8(b)). When there are 12 variables, this percentage is more than 25%. According to the experimental data, with 12 variables and each variable with the maximum domain size of 5, the average execution time of DT* approach is still less than 50 seconds.

Set 5: binary-valued arbitrary acyclic CP-nets. We vary s_{var} from 2 to 20. Similar to the results presented in Set 4, when the number of variables is large (more than 15) the average number of visited nodes of DT* algorithm is more than that of LVF algorithm (see Figure 9(a)). However, for binary-valued CP-nets in general, LVF does not guarantee completeness and the percentage of cases that the LVF algorithm returns incorrect answers is increasing as the number of variable increases (Figure 9(b)). When there are 20 variables, this percentage is more than 20%. While on average, DT* algorithm returns a correct answer to the given query in about 20 seconds.

Set 6: multi-valued arbitrary acyclic CP-nets. In the last set of experiments, we vary s_{var} from 2 to 10. The results with arbitrary acyclic CP-nets in multi-valued setting is similar to that in binary-valued setting (see Figure 10(a) and Figure 10(b)). When there are 10 variables, the percentage of incomplete cases the LVF algorithm is more than 20%; on the other hand, DT* guarantees to return a correct answer in about 9 seconds on average.

In summary, our experiments show that on average, our proposed DT* algorithm is much more efficient than the DF algorithm. It is as relatively efficient as LVF algorithm while guaranteeing soundness and completeness of the search process. From the experiment, we can also conclude that our proposed DT* algorithm allows dominance queries for CP-nets that are quite large and complex to be answered in reasonable time.

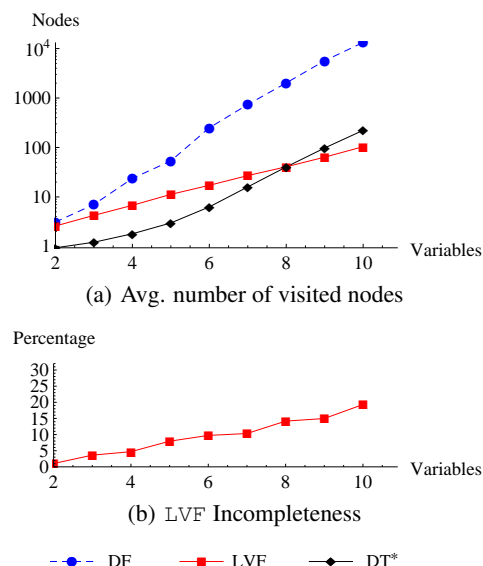


Figure 10: Multi-valued arbitrary acyclic CP-nets

5. CONCLUSION AND FUTURE WORK

In this paper, we have studied the problem of dominance testing in CP-nets. We have proposed a heuristic algorithm DT* for dominance testing with arbitrary acyclic CP-nets. The proposed approach significantly reduces the search effort without impacting soundness and completeness. We have also experimentally shown that the proposed algorithm is computationally efficient.

Nonetheless, the present work is only applicable for acyclic CP-nets. The investigation of techniques to deal with cyclic preferences need to be further explored.

6. ACKNOWLEDGEMENTS

This work is partially supported by the ARC Discovery Grant DP0987380.

7. REFERENCES

- [1] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Reasoning with conditional ceteris paribus preference statements. In *UAI*, pages 71–80. Morgan Kaufmann Publishers, 1999.
- [2] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. Preference-based constrained optimization with CP-nets. *Computational Intelligence*, 20:137–157, 2001.
- [3] C. Boutilier, R. I. Brafman, H. H. Hoos, and D. Poole. CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements. *Journal of Artificial Intelligence Research*, 21:135–191, 2004.
- [4] C. Domshlak, S. D. Prestwich, F. Rossi, K. B. Venable, and T. Walsh. Hard and soft constraints for reasoning about qualitative conditional preferences. *J. Heuristics*, 12(4-5):263–285, 2006.
- [5] J. Goldsmith, J. Lang, M. Truszczynski, and N. Wilson. The computational complexity of dominance and consistency in CP-nets. *J. Artif. Int. Res.*, 33(1):403–432, 2008.
- [6] C. Lafage and J. Lang. Logical representation of preferences for group decision making. In *KR*, pages 457–468, 2000.
- [7] J. Lang. Graphical representation of ordinal preferences: Languages and applications. In *ICCS*, pages 3–9, 2010.
- [8] F. Rossi, K. B. Venable, and T. Walsh. mCP nets: Representing and reasoning with preferences of multiple agents. In *AAAI*.
- [9] G. R. Santhanam, S. Basu, and V. Honavar. Dominance testing via model checking. In *AAAI*, 2010.

Distributed Problem Solving II

Resource-Aware Junction Trees for Efficient Multi-Agent Coordination

N. Stefanovitch
LIP6 - UPMC
75016 Paris, France
stefanovitch@poleia.lip6.fr

A. Farinelli
University of Verona
Verona, I-37134, Italy
alessandro.farinelli@univr.it

A. Rogers, N.R. Jennings
University of Southampton
Southampton, SO17 1BJ, UK
{acr,nrj}@ecs.soton.ac.uk

ABSTRACT

In this paper we address efficient decentralised coordination of cooperative multi-agent systems by taking into account the actual computation and communication capabilities of the agents. We consider coordination problems that can be framed as Distributed Constraint Optimisation Problems, and as such, are suitable to be deployed on large scale multi-agent systems such as sensor networks or multiple unmanned aerial vehicles. Specifically, we focus on techniques that exploit structural independence among agents' actions to provide optimal solutions to the coordination problem, and, in particular, we use the Generalized Distributive Law (GDL) algorithm. In this settings, we propose a novel resource aware heuristic to build junction trees and to schedule GDL computations across the agents. Our goal is to minimise the total running time of the coordination process, rather than the theoretical complexity of the computation, by explicitly considering the computation and communication capabilities of agents. We evaluate our proposed approach against DPOP, RDPI and a centralized solver on a number of benchmark coordination problems, and show that our approach is able to provide optimal solutions for DCOPs faster than previous approaches. Specifically, in the settings considered, when resources are scarce our approach is up to three times faster than DPOP (which proved to be the best among the competitors in our settings).

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: [Coherence and coordination, Multiagent systems]

General Terms

Algorithms, Performance, Experimentation

Keywords

multiagent coordination, junction tree, treewidth, variable elimination, heuristic algorithm, GDL, DCOP

1. INTRODUCTION

Many practical applications require the development of effective decentralised coordination techniques for cooperative multi-agent systems. For example, agent-based techniques have been widely

Cite as: Resource-Aware Junction Trees for Efficient Multi-Agent Coordination, N. Stefanovitch, A. Farinelli, A. Rogers and N. R. Jennings, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 363-370.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

used to control physical devices which can acquire and process information from the environment, such as sensor networks deployed to collect environmental data [12] or multiple unmanned aerial vehicles deployed to collectively patrol and map a defined area [16]. The development of decentralised coordination techniques is particularly challenging in these domains because such devices usually have constrained computational resources (due to the requirement of minimising power consumption) and because communication is usually limited in bandwidth and is dependent on the physical distance and mutual positions of the devices (due to the wireless communication technology frequently used). Moreover, to develop more cost effective systems, and to manage legacy, it is expected that the devices within such networks will be heterogeneous; having different computation and communication capabilities.

Recent work has shown that to develop effective and efficient coordination techniques it is crucial to exploit the structural independence between the agents' utility functions (i.e. the fact that the utility of each agent only depends on its own choice of action and that of a small number of locally interacting neighbours) [16, 14]. Doing so allows the decentralised coordination problem to be framed as a Distributed Constraint Optimisation Problem (DCOP), enabling a number of optimal algorithms to be used as solution techniques, e.g., ADOPT [9], OptAPO [8] and DPOP [15].

However, these algorithms take no account of the heterogeneous computational and communication resources available to the different agents within the system. In many settings, and particularly in the cooperative settings we focus on here, it may be beneficial to delegate computations such that (i) we take advantage of agents with greater than average computational capabilities, and (ii) we minimise communication between agents with poor communication links. Current algorithms for solving DCOPs do not consider such strategies. For example, DPOP arranges the constraint network into a pseudo-tree using a Depth First Search (DFS) method. While the DFS can be conveniently performed using distributed algorithms, it does not take into account agents' individual computation and communication capabilities, and it can result in an inefficient allocation of computations to the agents. In contrast Paskin and Guestrin developed an approach to cope with networks that have poor quality communication links (as it is frequently the case with wireless networks)[14]. Their approach uses the routing tree of the communication network to arrange agents into a *junction tree*, which is then further optimised in order to minimise communications. While this work takes computation and communication into account, it forces the junction tree structure to be a spanning tree of the communication network, which can, as we shall show later, significantly reduce the efficiency of the coordination process.

Thus, against this background, in this paper we address these shortcomings by proposing a *resource aware* solution technique for DCOPs. Our approach pre-processes the constraint network by

building a junction tree, over which optimal inference is performed using standard message passing techniques, such as those provided by the Generalised Distributive Law (GDL) framework [1]. Junction trees are well known structures, frequently used in graphical models and constraint processing [7] and while finding the optimal junction tree (in the sense of a minimal size of the largest clique) is NP-hard, a number of heuristics that build near optimal junction trees are well known. Here, we propose a distributed approach to build the junction tree that is based on the variable elimination algorithm [5], but is extended to consider the heterogeneous nature of both computational and communication resources within the network. In this context, the optimal tree is not the one that minimises the theoretical complexity of the computation (as is the case within the standard literature of junction trees), but is the one that minimises the total running time of the coordination algorithm (including both the time required by the agents to individually compute their partial solutions, and the time for these solutions to propagate up and down the junction tree).

In doing the above, this paper makes the following contributions to the state of the art:

- We present the first model of decentralised coordination that explicitly considers the total running time required by a coordination algorithm that operates in heterogeneous multi-agent system.
- We propose a novel distributed algorithm, based on the variable elimination algorithm, which uses a novel resource aware heuristic to minimise the running time of the coordination process (as defined above).
- We empirically evaluate the proposed technique on a simulated environment, comparing it with three state of the art approaches for multi-agent systems: the pseudo-tree built by DPOP, the junction tree formation algorithm of Paskin and Guestrin, and a benchmarking centralised approach. Our results show that, when communication resources are scarce, our resource aware heuristic improves upon previous techniques being up to 3 times faster than DPOP, which proved to be the best competitors in our settings.

The rest of the paper is organised as follows: Section 2 provides basic background knowledge on DCOPs and graphical models while Section 3 formalises the problem we are addressing. Section 4 details our approach and the resource aware heuristic we propose. Section 5 presents the empirical analysis of our approach, Section 6 discusses related work and Section 7 concludes.

2. BACKGROUND

We provide here a brief review of background knowledge concerning DCOPs, junction trees and the GDL framework.

2.1 Distributed Constraint Optimisation Problems

Formally a DCOP can be defined as a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \Psi \rangle$, where $\mathcal{A} = \{A_1, \dots, A_k\}$ is a set of agents, $\mathcal{X} = \{1, \dots, n\}$ is a set of variables. Each variable is owned by exactly one agent, but an agent can potentially own more than one variable. An agent is responsible for assigning values to the variables it owns. $\mathcal{D} = \{D_1, \dots, D_n\}$ is a set of discrete and finite variable domains, each variable i can take value in the domain D_i . Finally, $\Psi = \{\psi_1, \dots, \psi_m\}$ is a set of constraint functions that describe the constraints among variables. Each function $\psi_i : D_{i_1} \times \dots \times D_{i_{r_i}} \rightarrow \mathbb{R}$ depends on a set of variable $\mathbf{X}_i \subseteq \mathcal{X}$, where $r_i = |\mathbf{X}_i|$ is the arity of the function. Each function assigns a real value to each possible assignment of

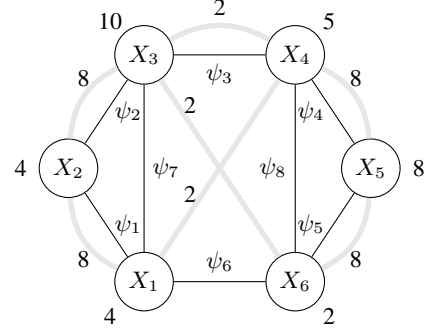


Figure 1: Example of a MAS coordination problem : communication (wide grey edges) and constraint network (thin black edges), nodes represent both agents and variables

the variables it depends on. Given this settings, we wish to find the variable assignment \mathbf{X}^* such that the sum of all constraint functions is maximised:

$$\mathbf{X}^* = \arg \max_{\mathbf{X}} \sum_{i=1}^m \Psi_i(\mathbf{X}_i) \quad (1)$$

2.2 Junction Trees

DCOPs can be solved using message passing algorithms across the *constraint network*; a graph representation of a DCOP where nodes are variables and edges are constraint functions. However, in order to ensure completeness and termination, the constraint network must be a tree. If this is not the case, then it is necessary to transform the original constraint network into a special graphical structure called a junction tree. This is done by forming a clique graph whose nodes (cliques) and edges (separators) are clusters of variables. A junction tree is simply a clique graph which satisfies the following four properties: single-connectedness, running intersection, covering and maximality. Single-connectedness ensures that the graph is a tree, yielding termination. Running intersection ensures that any variable present in the intersection of the domain of two cliques is also present in every clique of the path joining them, yielding correctness. Covering ensures that the domain of every constraint function of the DCOP is the subset of at least one clique. Maximality states that a clique can not have a domain which is a subset of the domain of another clique. One of the most well-known algorithms to transform a constraint graph into a junction tree is the variable elimination algorithm. This algorithm works by sequentially selecting variables of the constraint graph, eliminating them and forming cliques accordingly (see [7] for more details).

2.3 GDL

Having formed a junction tree, the optimal solution of the DCOP can be found using a suitable message passing algorithm, of which GDL is the most general [1]. The GDL algorithm uses two operators, \otimes for function combination and \oplus for function marginalisation, and exploits the distribution property of \otimes over \oplus . It works by passing messages along the edges of the junction tree and performing computation at the level of the nodes. DCOPs can be solved by GDL using the *sum* and *max* operators respectively.

A message from a clique n to a clique m is a utility function defined recursively over the intersection of the domains of n and m

by the formula: $\forall \mathbf{y} \in d(m) \cap d(n)$, where the function d gives the set of variables associated to a clique or separator (collect phase):

$$\psi_{n \rightarrow m}(\mathbf{y}) = \bigoplus_{\mathbf{x} \in d(n) \setminus d(m)} (\bigotimes_{i \in \Gamma(n) \setminus m} \psi_{i \rightarrow n} \otimes \psi_n)(\mathbf{x} \cup \mathbf{y})$$

A special node, called the root, receives the information from all of its neighbours. It computes the optimal instantiation of its variables and then starts the recursive phase of local optimisation yielding the global optima (propagation phase). Specifically when a clique m receives the instantiation from its parent n it conditionally optimise the instantiation of its variable and then propagates the set of instantiation it knows:

$$\mathbf{x}_n^* = (\arg \bigoplus_{\mathbf{x} \in d(n) \setminus d(m)} (\bigotimes_{i \in \Gamma(n) \setminus m} \psi_{i \rightarrow n} \otimes \psi_n)(\mathbf{x} \cup \mathbf{x}_m^*)) \cup \mathbf{x}_m^*$$

Both the collect and propagation phases can be performed with a linear number of messages. However, during the collect phase, the computation of the messages and the size of the messages are exponential in the size of the clique sending them, and the cardinality of the corresponding separator respectively.

The use of junction trees in combination with the GDL algorithms is attractive as they inherently work in a distributed way by message passing. Moreover such an approach is efficient as it is exponential only in the treewidth rather than in the total number of variables. The treewidth is a parameter of a tree decomposition which is the size of the largest clique, and is usually far smaller than the total number of variables. Finding a junction tree of minimal treewidth is however an NP-hard problem [7].

3. PROBLEM DESCRIPTION

Having presented the necessary background, we now formally describe the problem that we tackle; that of, minimising the total running time of a coordination algorithm when faced with heterogeneous computational resources, and a bandwidth constrained communication structure. To this end, we introduce the concept of a *computational* task to model the GDL solution process for DCOP. A computational task $\tau(\psi_{n \rightarrow m}(\mathbf{y}))$ describes the amount of computation that an agent has to perform in order to *compute* the message $\psi_{n \rightarrow m}(\mathbf{y})$ defined in Section 2.3.

Since the computation of a message $\psi_{n \rightarrow m}(\mathbf{y})$ recursively depends on the computation of messages from neighbouring cliques, these computational tasks are tied by the set of execution constraints $EC(\tau_i) = \{\tau_1^i, \dots, \tau_k^i\}$ where τ_j^i are tasks that must be executed before τ_i . We denote the set of all the computational tasks as \mathbb{T} , and this consists of the set of cliques of the junction tree and the set of constraint functions¹. Execution of a computational task involves the processing of constraint functions and received messages (i.e., the summation of those functions and the maximisation over some subsets of the decision variables) in order to compute the message. Each computational task is assigned to a single agent, which is responsible for all the aspects of the execution of this task. We denote $\alpha : \mathbb{T} \rightarrow \mathcal{A}$ the function representing this allocation.

The *completion time* of a task τ depends on its size (given by the *size* function), on the computational power of the agent $\alpha(\tau)$ expressed as the number of constraint checks per unit of time (given by the *speed* function), and on the characteristics of the communication links used to route the messages produced by the execution of $EC(\tau)$ to the agent $\alpha(\tau)$ (given by the *trans* function). The completion time of a single computational task, can then be defined

¹Constraint functions can be seen as computational tasks requiring no processing from the agent side, but requiring some non-zero transmission time to the agent responsible of the clique to which the constraint function is allocated.

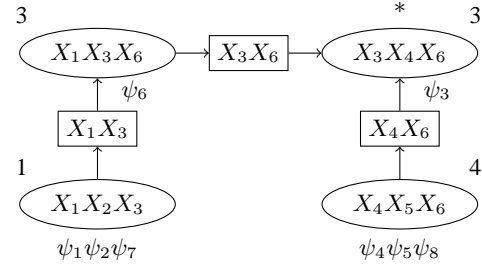


Figure 2: Instance of a solution obtained on the MAS coordination problem of Figure 1

as a function $CT : \mathbb{T} \rightarrow \mathbb{N}$, whose expression is:

$$CT(\tau) = \max_{\tau_i \in EC(\tau)} CT(\tau_i) + \text{trans}_{\alpha(\tau_i), \alpha(\tau)}(\tau_i) + \text{size}(\tau) / \text{speed}(\alpha(\tau))$$

Note that, we assume that agents are not multitasking and tasks are not preemptive, such that agents can either compute, send or receive messages at any time, and can not interrupt one of those activities if already started. We consider a restricted communication structure composed of pairwise communication links. For each link we consider a symmetric limited bandwidth that defines the time required to transmit messages over the link. If an agent on the shortest path between two agents is performing one of the above activities, messages between these two agents have either to wait or to find a new route. Such a setting models some important aspects of wireless sensor networks such as limited bandwidth, limited connectivity, multi-hop communication and possible network congestions. Small bandwidth values can represent both low throughput reliable communication links or high throughput unreliable communication links.

Figure 1 shows an exemplar instance of a constraint network associated with a restricted communication network. The nodes represent decision variables, the thin black edges are constraint dependencies between the variables. Constraints that hold between variables are associated with constraint functions ψ_i as shown in the figure. The thick grey edges represent the underlying communication network between the agents responsible for the variables. Numbers next to those edges correspond to the bandwidth of the communication links while numbers next to the agents represent computational speed of the agents.

Now, given any specific constraint network, we can define the set of computational tasks and the set of messages that need to be computed and propagated according to the GDL algorithm. The GDL algorithm can then be described as the execution of the associated computational tasks using agents as computational resources. Specifically, given a DCOP instance $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \Psi \rangle$, to obtain the set of associated computational tasks we have to choose a set \mathcal{C} of cliques, an allocation $\alpha_\Psi : \Psi \rightarrow \mathcal{C}$ of constraints functions to cliques, and an allocation $\alpha_C : \mathcal{C} \rightarrow \mathcal{A}$ of the cliques onto the agents. We can now define $MS(\mathcal{C}, \alpha_\Psi, \alpha_C)$ as the time to complete all tasks subject to the execution constraints. Such a criterion is known as the makespan in the scheduling literature², and in our setting it corresponds to the time at which the last task has been computed (which is the maximisation of the root clique). Therefore, representing the special task associated to the root clique of the GDL algorithm as τ_r , the makespan can be defined as: $MS =$

²The makespan is a concept more general but akin to the number of non concurrent constraint checks in DCOP literature since it explicitly takes into account communication delays

$CT(\tau_r)$. Our aim is then to find the junction tree, the allocation of constraints to cliques and cliques to agents that will minimise this makespan, such that:

$$\arg \min_{C, \alpha_\Psi, \alpha_C} MS(C, \alpha_\Psi, \alpha_C)$$

Unfortunately, optimally scheduling a set of tasks onto a set of heterogeneous processors, even without considering communication, is known to be NP-hard [18]. Therefore, in this paper, we aim to design an effective heuristic that works well in practise.

4. RESOURCE-AWARE JUNCTION TREES

Here we present our approach to build junction trees and allocate computational tasks to agents in order to heuristically minimise the makespan of the coordination process. We first note that the minimisation problem stated in the previous section can be divided into two sub-problems: (i) finding a suitable junction tree decomposition of the problem, by defining cliques and allocating constraint functions to cliques; (ii) allocating the resulting computational tasks to agents to minimise the makespan. However, the two subproblems are interconnected because grouping of variables in cliques induced by the junction tree impacts on the computation and communication that agents need to perform. The approach we propose treats both subproblems at the same time, and it is divided into two key parts: a distributed protocol that implements variable elimination and a novel resource aware heuristic (RAH) that seeks to select variables in order to heuristically minimise the makespan of the coordination process.

Distributed Protocol for Variable Elimination

This protocol is a distributed negotiation protocol that extends the variable elimination algorithm by making use of calls for proposals (CFP) and bids in order to determine the next variable to eliminate and the agent responsible for the associated clique maximisation. Our protocol proceeds by each agent broadcasting one CFP for each variable it is responsible for. Each agent then replies to a CFP by estimating the makespan associated with itself being responsible for the computation of the clique that would be created if the variable specified in the CFP was eliminated. Given this description, we now formally define the key elements of our protocol.

- A *constraint* is a tuple $\langle f, \mathcal{X}_f, \psi_f, a_f \rangle$, where f is the identifier of the constraint, $\mathcal{X}_f \subseteq \mathcal{X}$ is the domain (or label) of the constraint function, $\psi_f : 2^{\mathcal{X}_f} \rightarrow \mathbb{R}$ is the constraint function and $a_f \in \mathcal{A}$ is the identity of the agent owning the constraint.

- A *variable* is a tuple $\langle X, \Gamma_X, \Gamma_X^{past}, a_X, c_X \rangle$, where X is the identifier of the variable, $\Gamma_X \subseteq \mathcal{X}$ is the set of neighbours of this variable in the constraint graph, $\Gamma_X^{past} \subseteq \mathcal{X}$ is the set of former neighbours that have already been eliminated, $a_X \in \mathcal{A}$ is the agent responsible for this variable and $c_X \in \mathcal{C}$ is the clique related to the elimination of this variable (initially void). For instance, in the Figure 1 variable X_2 is represented by the following tuple $\langle X_2, \{1, 2, 3\}, \{\}, 2, \emptyset \rangle$.

- A *clique* is a tuple $\langle c, \mathcal{X}_c, \Psi_c, \Gamma_c^C, X_c, a_c \rangle$, where c is the identifier of the clique, $\mathcal{X}_c \subseteq \mathcal{X}$ is the domain (or label) of the clique, $\Psi_c \subseteq \Psi$ is the set of constraint functions allocated to the clique (initially void), $\Gamma_c^C \subseteq \mathcal{C}$ is the set of neighbours in the junction tree (initially void), $X_c \in \mathcal{X}$ is the variable whose elimination led to the creation of c and $a_c \in \mathcal{A}$ is the identity of the agent responsible of the clique. For instance in the Figure 2 the clique $X_1X_2X_3$ is represented by the following tuple $\langle X_1X_2X_3, \{X_1, X_2, X_3\}, \{\psi_1, \psi_2, \psi_7\}, \{X_1X_3X_6\}, X_2, 1 \rangle$.

- An *agent* is a tuple $\langle a, \mathcal{X}_a, \Psi_a, \mathcal{C}_a, \Gamma_a^{com} \rangle$, where a is the identifier of the agent, $\mathcal{X}_a \subseteq \mathcal{X}$ is the subset of variables the agent owns, $\Psi_a \subseteq \Psi$ is the set of constraint functions allocated to the

agent, $\mathcal{C}_a \subseteq \mathcal{C}$ is the set of cliques allocated to the agent (initially void) and $\Gamma_a^{com} \subseteq \mathcal{A}$ is the set of neighbours of the agent in the communication graph. For instance in the Figure 2, agent 3 is represented by the following tuple $\langle 3, \{X_3\}, \{\psi_3, \psi_6\}, \{X_1X_3X_6, X_3X_4X_6\}, \{1, 2, 3\} \rangle$.

- A *CFP* is initiated by one agent which proposes to another to take the responsibility of a clique (which is only *partially* created and initialised). Formally a CFP is a tuple $\langle s, c, d \rangle$, where $s \in \mathcal{A}$ is the sender agent, d is a date (the number of the bidding turn), and c is a clique, whose function (of exponential size) is not created. The initialised fields of c are Ψ_c - with the list of subfunctions that would be allocated to c , Γ_c^C - with the list of neighbouring cliques in the junction tree (this set is needed to determine the set of separators, which in turns define the scope of messages that would have to be transmitted or received by c and X_c - which identifies the variable for which the CFP has been sent.

- A *bid* is formally a tuple $\langle s, r, cfp, clique, v \rangle$ where $s \in \mathcal{A}$ is the sender agent, $r \in \mathcal{A}$ is the addressee agent, cfp is the CFP for which the bid is a reply, *clique* is the identity of the clique to which the constraint functions of the clique of the CFP will be allocated and $v \in \mathbb{R}^+$ is an evaluation of the time it would take the agent s to compute the clique proposed by agent r in his CFP.

Algorithm 1 Variable selection and elimination (*selection*)

```

1: parallel { treat_CFP() }
2: date ← 0
3: while date < |X| do
4: // compute variables' heuristic values
5: send_CFP()
6: treat_bids()
7: // select the next variable to eliminate
8: selected_bid ← consensus_select(best_bid)
9: c ← selected_bid.cfp.c
10: X ← X_c
11: // allocate the clique
12: if selected_bid.s = a then
13:   C_a ← C_a ∪ {c_X}
14: end if
15: // update the set of constraint
   functions
16: if X ∈ X_a then
17:   Ψ_a ← Ψ_a ∪ Ψ_c
18: end if
19: // add new constraint dependencies
   between each pair of neighbours
20: for Y ∈ X_a ∩ X_c do
21:   for Z ∈ X_c : Y ≠ Z do
22:     if Y ∉ Γ_Z then
23:       // extend variables' neighbourhood
24:       Γ_Z ← Γ_Z ∪ {Y}
25:       Γ_Y ← Γ_Y ∪ {Z}
26:     end if
27:   end for
28: end for
29: // update variables neighbourhood with
   respect to X
30: for all X_a ∈ X_a do
31:   // remove variables' constraint edges
32:   Γ_X_a ← Γ_X_a \ {X}
33:   // memorise variables' past constraint
   edges
34:   Γ_X_a^{past} ← Γ_X_a^{past} ∪ {X}
35: end for
36: // reset the algorithm's local variables
37: best_bid ← ∅
38: date ← date + 1
39: end while
40: compute a maximum spanning tree and connect the cliques ac-
   cordingly
41: start the GDL message passing inference algorithm

```

Having defined our terms we now present our protocol in algorithms 1-5. Variable elimination works first by computing a heuristic evaluation for each variable. This is done by concurrently executing the functions *treat_CFP* (line 1 of Algorithm 1), *send_CFP* and *treat_bids* (lines 5-6 of Algorithm 1). CFP are used in order to assess the impact on the makespan of the elimination of a variable. In order to do so a CFP contains the clique corresponding to the elimination of this variable. No clique is added to the junction tree until a consensus has been reached on the variable to eliminate (lines 9-14 of Algorithm 1), at which time only one clique is actually created.

- *send_CFP* computes and sends a CFP for each of the variable an agent is responsible for. The field Ψ_c is filled with the set of constraint functions³ that would be allocated to this clique if actually created, and the set Γ_c^C is filled accordingly with the neighbourhood of this clique. Notice that this is a conservative estimation as the actual neighbours will be computed in Algorithm 1 (line 40). A CFP is compiled only with the information available in the direct neighbourhood of the variable in the constraint graph (lines 4, 6 and 8). The CFP is then propagated to all the agents.⁴

- *treat_CFP* waits for incoming CFP and calls upon receiving the procedure *RAH_evaluate_clique*. This function returns an heuristic evaluation of the impact on the makespan for this agent to compute the clique in the CFP. If the clique of the CFP has a domain which is a subset of one of the cliques associated with a variable of the agent, the evaluation is then set to 0, and the clique that would be created is set to this already existing clique. Otherwise, the value is left untouched and the clique that would be created is the one in the CFP (lines 4-9). This is done in order to enforce the use of maximal cliques only. The bid is then sent to the sender agent.

- *treat_bids* handles received bid messages. If an incoming bid has an evaluation lower than the current best bid, it is selected as the best bid (lines 5-8). The agent corresponding to the lowest evaluation is stored inside the bid data structure. When all the bids have been received, the flow of control returns to Algorithm 1.

- *selection* is the function implementing the actual distributed variable elimination algorithms. Once the previous algorithms complete, it selects a variable to eliminate according to the heuristic evaluation computed (line 7). This step is made through a consensus, where each agent proposes the best bid it has received. Various algorithms such as a wave propagation algorithm [17] could be used in order to perform a consensus, however the simplest way is to propagate each message to all the agents. The variable and the clique are then extracted (lines 8-9). The agent selected by the winning bid adds the new clique to the set of cliques that it is responsible for (lines 11-13). The agent owning the selected variable first updates the set of its constraint functions by removing the ones allocated to the new clique (line 17). All the neighbouring agents of the selected variables update concurrently both their neighbourhood (lines 31-34) and the constraint neighbourhood of their variables (lines 24-25). It is necessary to do so in order to store both the deleted edges (used to compute the heuristic) and the full set of dependencies between variables (used to communicate with all the relevant agents). When all the variables have been eliminated, the cliques are connected together using a maximum spanning tree, which enforces the tree structure and the running intersection property of the junction tree [3]. This can be done efficiently in a distributed way using the approach of [6]. Finally, Algorithm 1 starts

³The actual function is not transmitted as it is of exponential size. The heuristic can be computed by considering only its domain (see line 13 of Algorithm 5)

⁴While each agent might not be able to reach all other agents directly, messages will be propagated to all the agents possibly through multi-hop communication.

the GDL messages-passing algorithm (line 41). The root node is selected as the one in the middle of the diameter of the junction tree.

Algorithm 2 Outgoing CFP management (*send_CFP*)

```

1: // compute and send CFPX
2: for all  $X \in \mathcal{X}_a$  do
3:   // compute the domain of the clique
4:    $\mathcal{X}_c \leftarrow \{X\} \cup \Gamma_X$ 
5:   // compute the set of related constraint
   functions
6:    $\Psi_c \leftarrow \cup_{Y \in \Gamma_X \cup \Gamma_X^{past}} \{ \langle f, \mathcal{X}_f, \emptyset, a_Y \rangle : \psi_f \in \Psi_{a_Y}, \mathcal{X}_f \subseteq \mathcal{X}_c \}$ 
7:   // compute the set of related cliques
8:    $\Gamma_c^C \leftarrow \cup_{Y \in \Gamma_X^{past}} \{c_Y\}$ 
9:   // partially create the clique
10:   $clique \leftarrow \langle c, \mathcal{X}_c, \Psi_c, \Gamma_c^C, X, \emptyset \rangle$ 
11:  // create the CFP
12:   $CFP_X \leftarrow \langle a, clique, date \rangle$ 
13:  broadcast CFPX
14: end for

```

Algorithm 3 Incoming CFP management (*treat_CFP*)

```

1: if  $receive(cfp = \langle s, c, d \rangle) \wedge d = date$  then
2:   $v \leftarrow RAH\_evaluate\_clique(c)$ 
3:  // enforce the use of maximal cliques
   only
4:  if  $\exists X \in \mathcal{X}_a : \mathcal{X}_c \subseteq \mathcal{X}_{c_X}$  then
5:     $clique \leftarrow c_X$ 
6:     $v \leftarrow 0$ 
7:  else
8:     $clique \leftarrow c$ 
9:  end if
10:  $bid \leftarrow \langle a, s, cfp, clique, v \rangle$ 
11: send bid
12: end if

```

Algorithm 4 Incoming bid management (*treat_bids*)

```

1:  $bids \leftarrow \emptyset$ 
2: while  $|bids| < |\mathcal{A}|$  do
3:  if  $receive(bid = \langle s, r, cfp, clique, v \rangle) \wedge r = a \wedge cfp.d = date$  then
4:     $bids \leftarrow bids \cup \{bid\}$ 
5:    if  $best\_bid.v > v$  then
6:       $best\_bid \leftarrow bid$ 
7:       $v \leftarrow 0$ 
8:    end if
9:    // select the next variable
10:  end if
11: end while

```

As there are a finite number of variables and one variable is always eliminated at the end of each turn, this algorithm will always terminate provided that there is no message loss. While we do not deal with such an issue here, we note that such issue could be addressed by using other consensus approaches [4] or specific communication protocol (such as for example TCP). Our protocol is fully distributed as an agent only needs to know the constraints it is involved with and the total number of agents in the system, at no moment does an agent know the full set of constraints or variables. In contrast, each agent executes the part of the variable elimination relative to their variables. The consensus protocol ensures that at each step, all the agents are synchronised on the identity of the eliminated variable. Therefore, our protocol is correct, and results in the same junction tree that would be created through a conventional centralized variable elimination using our resource aware heuristic.

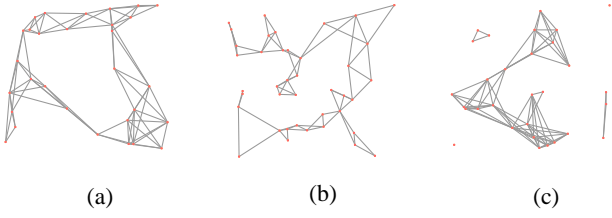


Figure 3: Constraint graphs of (a) ring, (b) tree and (c) cluster instances

Resource Aware Heuristic

Algorithm 5 RAH heuristic (*RAH_evaluate_clique*(c : clique))

```

1:  $v \leftarrow X_c$ 
2:  $eval \leftarrow +\infty$ 
3:  $a \leftarrow$  identity of the current agent
4:  $timeComp \leftarrow s(d(c))/speed(a)$ 
5: // estimate trans for  $\tau \in \mathcal{C}$  (separators)
6:  $timeSep \leftarrow 0$ 
7: for all  $c' \in \Psi_c$  do
8:    $timeSep \leftarrow \max(timeSep, sp(a, a_{c'}, s(\mathcal{X}_c \cap \mathcal{X}_{c'})))$ 
9: end for
10: // estimate trans for  $\tau \in \Psi$  (constraint functions)
11:  $timeSub \leftarrow 0$ 
12: for all  $f \in \Gamma_c^c$  do
13:    $timeSub \leftarrow \max(timeSub, sp(a, a_f, s(\mathcal{X}_f)))$ 
14: end for
15:  $eval \leftarrow \min(eval, timeComp + timeSep + timeSub)$ 
16: return  $eval$ 

```

The *RAH_evaluate_clique* procedure takes as input a clique and gives an heuristic estimate of the impact on the makespan of the computation of the clique on the current agent. This is done by greedily allocating constraint functions to this clique and allocating the clique to one agent. In more detail, this procedure computes the sum of three values: the time to compute the task, the time to transfer the allocated constraint functions, and the time to transfer the messages of the execution constraints (lines 5-14 of Algorithm 5). The evaluation of the time to transfer messages is computed as the size of the message divided by the maximal bandwidth between the involved agents. This computation is done by the *sp* function (lines 8, 13 of Algorithm 5), where the function *s* returns the number of elements in the utility table representing a function given its domains. Note that this is an approximation since congestion in the communication network will impact this result, however as discussed in Section 3 we focus here on an effective heuristic approach rather than an optimal allocation which is known to be NP-hard [10].

Figure 2 depicts the results of our approach applied to the MAS coordination problem of Figure 1. The constraint function allocation are indicated beneath the cliques, and clique allocation to agents are indicated on top of each clique with the number of the responsible agent. The asterisk denotes the root. In this instance, agent 3 is responsible for two cliques as the heuristic estimates that the makespan could best be reduced by saving communication rather than exploiting distribution of computations.

5. EMPIRICAL EVALUATION

We empirically evaluate our RAH algorithm against two closely related state of the art distributed inference algorithms: (i) DPOP, whose pseudo-tree is built with a distributed DFS approach⁵, and

⁵We use here the most connected node (MCN) heuristics, which as

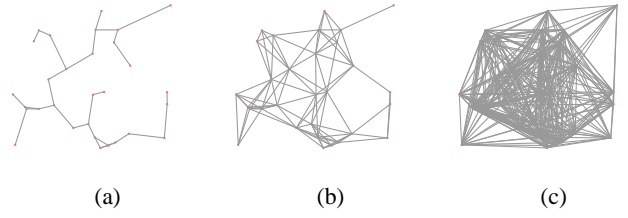


Figure 4: Three communication graphs of the cluster instance, where (a) $r=100$, (b) $r=300$ and (c) $r=700$

(ii) RDPI (Robust Distributed Probabilistic Inference), the initial junction tree construction of Paskin and Guestrin which builds a junction tree over the minimum spanning tree of the communication network [14]. We also compare against a centralised benchmarking approach, which generates a near-optimal junction tree using the standard variable elimination algorithm and the *minimum size* (MS) heuristic, and then allocates all tasks to the fastest agent in the system. We refer to this algorithm as MS.

We benchmark these algorithms on a set of three scenarios with different constraint network structures. Agents are located in a square of fixed size of 1000 unit and we consider three different topologies: rings, trees and clusters with 30, 40 and 30 agents respectively. For simplicity we consider that there are as many variables as agent; constraints are *n*-ary. Figure 3 shows the structure of the three constraint networks considered.

For each scenario we perform a set of experiments by varying the communication range (i.e., the distance within which two agents can communicate) in order to study the impact of the availability of communication resources on the coordination procedure. The communication range describes the availability of the communication resources. A communication range lower than 100 indicates that only a few communication links between neighbouring agents exist, while a communication range greater than 500 implies that each agent is roughly connected to at least half of the agents in the system. The wider is the communication range, the more likely it is to find direct high bandwidth communication links between any two agents. In the limit, such a case is equivalent to having no communication constraint at all.

Figure 4 represents the structure of the communication network for three different ranges of communication in the *cluster* instance. For each experiment (i.e. a fixed constraint and communication network structure) the agents' computational speed and links' bandwidth are randomly drawn from the set $\{2,4,8\}$.

Results

We measure the makespan and the *treewidth* (the size of a maximal clique [7]) for each algorithm. The makespan is empirically computed on a simulation environment matching the full characteristics described in Section 3 (i.e., blocking communications, non multitask agents and non preemptive tasks) using the same routing policy for all the algorithms (except RDPI which has its own). Note that in these simulations we include the full effect of network congestion. The unit of makespan measurement is the time step of the simulator. For each experiment we performed 15 runs, and we report the mean and the standard error of the mean in Figure 5. The treewidth measurements are reported in Table 1. In both case *r* notes the communication range parameter.

In the *ring* and *cluster* scenario, the resource aware heuristic performs up to three times faster than DPOP when resources are discussed in [15] drives the DFS to obtain pseudo-trees with low treewidth

scarce (communication range below 100). The performance then stabilises (two times faster than DPOP) when the communication range increases. In the *tree* scenario, because the communication and constraint networks are both tree structured and generated according to a distance measure between agents, communication along the DPOP pseudo-tree matches the communication network more closely than in other instances, where the agent responsible for neighbouring cliques are less likely to be neighbours in the communication graph. As a result the performances of RAH and DPOP are similar until the point where communication is no longer a scarce resource (communication range of 400) where DPOP is able to perform better. These results show the importance of taking into account the differences between the communication and constraint networks in DCOPs.

The comparison between RAH and MS shows that the difference in performance is not only highly contrasted but is reversed when communication is no longer a scarce resource. Specifically in such a case MS is able to perform better than the resource aware heuristic we propose as all the agents are able to directly communicate with fast communication links with the centralising agent, indicating that in this case centralising the solution is more efficient. Conversely when resources are highly constrained the RAH is able to perform up to 3 times faster than MS.

The makespan obtained with the RDPI algorithm was extremely high (around two orders of magnitude higher than MS) in our settings, and are thus not reported in the Figure 5. This is related to the high treewidth (see Table 1) of the junction trees which is up to six times the treewidth of a near-optimal junction tree and two to three times larger on average. This is due to the fact that RDPI forces the junction tree to be built on top of a spanning tree of the communication network and this can result in junction trees with very large treewidth. In order to tackle this problem, Paskin proposes to use simulated annealing in order to optimise the junction tree. However, such a procedure requires an expensive distributed evaluation procedure in order to evaluate the cost of a local move and an unbounded number of messages [13]. As we focus here on the efficiency of junction trees that can be obtained with simple preprocessing techniques, we only report for RDPI the performance of the initial junction tree. The reported experimental evidences suggest that the cost of RDPI when the optimisation procedures converge, is within a factor of two of hypothesized optimal junction tree, which was built using an off-line centralised procedure, while the initial tree is up to seven times worse than that. However, notice that in our experiments RDPI results were orders of magnitude worst than competitors.

Furthermore notice that the treewidth for MS and RAH are very similar (see Table 1) but RAH clearly outperforms MS when communication is scarce (see Figure 5). These results again show the importance of taking into account agents' communication and computation capabilities when building the junction tree.

Summarising, our results show that while the treewidth of the junction tree remains an important parameter as it has an exponential impact on the efficiency of the algorithm, junction trees with higher treewidths can still result in better overall performances in such heterogeneous distributed settings if computations are appropriately scheduled across agents.

Complexity

While the running time depicted in Figure 5 only shows the relative performance of the different junction trees, it is important for real-world applications to also take into account the distributed running time of the preprocessing steps of all those algorithms. We discuss here the complexity, in terms of number of messages exchanged for the different approaches. For ease of notation, let us assume there

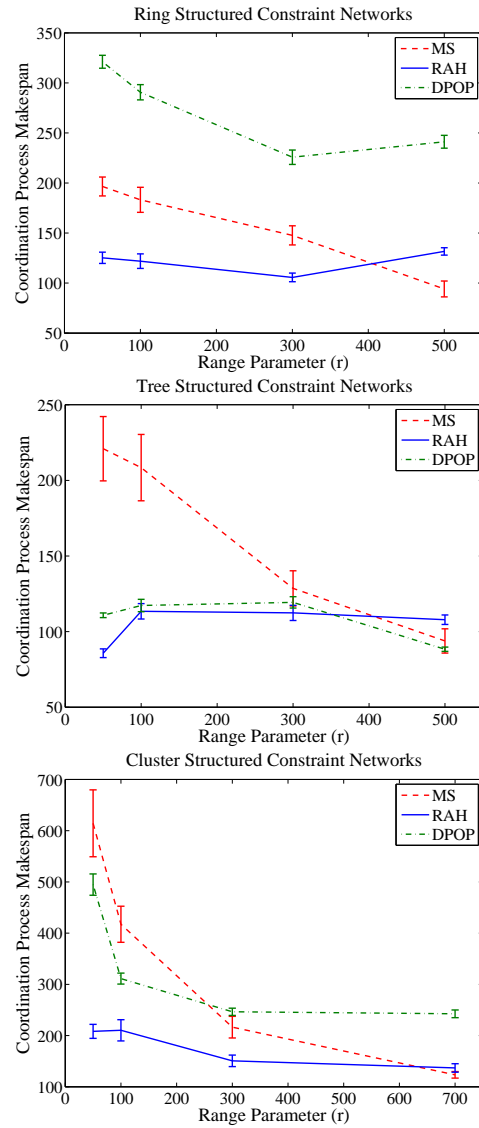


Figure 5: Coordination process makespan for the a) ring b) tree and c) cluster structured constraint graph instance

	r	MS	RAH	DPOP	RDPI
<i>cycle</i>	50	5	5.8 ± 0.1	6	10.4 ± 0.3
	100	5	5.4 ± 0.1	6	8.8 ± 0.2
	300	5	5.2 ± 0.1	6	11.1 ± 0.5
	500	5	5.1 ± 0.1	6	19.4 ± 0.5
<i>tree</i>	50	4	4.4 ± 0.1	5	13.3 ± 0.7
	100	4	4.4 ± 0.1	5	18.5 ± 0.7
	300	4	4.2 ± 0.1	5	20.6 ± 1.0
	500	4	4.1 ± 0.1	5	20.6 ± 1.0
<i>cluster</i>	50	7	7.0 ± 0	7	8 ± 0
	100	7	7.4 ± 0.1	7	8 ± 0
	300	7	7.1 ± 0.1	7	13 ± 0.7
	700	7	7.0 ± 0	7	24.2 ± 0.5

Table 1: Benchmarked treewidths

are as many agents as variable, where n is this number, tw is the treewidth, and that each agent possesses exactly one variable.

In terms of number of messages, DFS exploration uses $2n$ messages, and the MCN heuristic uses tw messages at each steps, yielding a number of messages for our DPOP implementation in $O(n tw)$. RDPI uses $O(n \log n)$ messages in order to build a spanning tree and then $2(n-1)$ messages in order to build the junction tree on top of it, yielding a number of messages in $O(n \log n)$. The MS algorithm is centralised and therefore each agent sends to the centralising agent its information regarding the variables and variables neighborhood, requiring the exchange of n messages. Finally the number of messages of our approach is the following. During the step $0 < k < n$ of Algorithm 1 $n-k$ CFP and $n(n-k)$ bids are sent, yielding a total number of messages for RAH in $O(n^3)$.

While the number of messages of our algorithm is higher than the others, as the results show, our approach can yield better running time for the DCOP solving algorithm for the solution phase. For real applications the measure we are interested in is the combined running time of the preprocessing phase and the actual solution phase. Such a running time depends on various parameters, the number of variables n , the tree-width tw and the cardinality d of the variables and also depends on the computation and communication capabilities of the multi-agent system. Now, the number of messages of the preprocessing phase for RAH is higher than competitors, but notice that messages sent in this preprocessing phase are of fixed size with respect to tw and d , while the complexity of the junction tree solution phase is exponential in tw with a basis of d . Therefore, depending on the values of the above parameters, the time required to send the messages for the preprocessing phase can be negligible with respect to the gain obtained in the running time for the solution phase. For instance if we consider the *cluster* experiment, we have $n = 30$, $tw = 7$, $d = 2$. The maximal time to compute a clique in such a case is $2^7 = 128$ times steps, while the RAH algorithm needs to exchanges $27 \cdot 10^3$ messages. However, if we consider $d = 10$, the complexity of the junction tree solution phase become prevalent with $10 \cdot 10^6$ times steps while the number of messages sent by RAH does not change.

Thus, depending on the settings of a coordination problem our algorithm can provide substantial gains in terms of total running time despite having a preprocessing overhead greater than the ones currently used in DPOP, MS and RDPI.

6. RELATED WORK

The use of junction trees (and other related graphical models) for solving DCOPs and the development of distributed approaches for junction tree compilation is a recent research topic that is gaining increasing attention. For example, Xia and Prassana use a distributed junction tree creation algorithm based on a DFS tree and propose to select the root so as to minimise the makespan [19], Otten and Dechter propose an heuristic for graphical models based on problem size measure that aims at load balancing efficiently the junction tree inference on as set of processors [11], Allouche and al. explicitly consider the problem of using distributed variable elimination in order to solve hard constraint optimisation problems [2]. However, none of these approaches address the problem from a MAS perspective, and as such they do not consider heterogeneity of computation and communication and they do not focus on having a distributed approach

7. CONCLUSION

In this work we take a first important step to explicitly consider multi-agent system specific issues (such as heterogeneity of computation and communication across the agents) when applying so-

lution techniques developed in the graphical model community to decentralised constraint optimisation.

Specifically, we show the importance of taking into account the actual resources of a multi-agent system when solving combinatorial optimisation problems across it, and validate our approach on benchmark coordination problems

Future work are divided in two directions. The first is to empirically validate our approach on a deployed wireless sensor network. The second aims to investigate bounded approximate algorithms. Addressing the trade-off among communication, computation and the bound that can be provided on solution quality.

8. REFERENCES

- [1] S. Aji and R. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.
- [2] D. Allouche, S. de Givry, and T. Schiex. Towards parallel non serial dynamic programming for solving hard weighted csp. In *Principles and Practice of Constraint Programming*, volume 6308 of *Lecture Notes in Computer Science*, pages 53–60, 2010.
- [3] J. R. S. Blair and B. Peyton. An introduction to chordal graphs and clique trees. *Institute for Mathematics and Its Applications*, 56, 1993.
- [4] T. D. Chandra and S. Toueg. Unreliable failure detectors for reliable distributed systems. *J. ACM*, 43(2):225–267, 1996.
- [5] R. Dechter and J. Pearl. Network-based heuristics for constraint-satisfaction problems. *Artif. Intell.*, 34(1):1–38, 1987.
- [6] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. Program. Lang. Syst.*, 5(1):66–77, 1983.
- [7] K. Kask, R. Dechter, J. Larrosa, and A. Dechter. Unifying tree decompositions for reasoning in graphical models. *Artif. Intell.*, 166(1-2):165–193, 2005.
- [8] R. Mailler and V. Lesser. Solving distributed constraint optimization problems using cooperative mediation. In *Proc. of the 3rd Int. Conf. on Autonomous Agents and MultiAgent Systems*, pages 438–445, 2004.
- [9] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, (161):149–180, 2005.
- [10] A. Moukrim and A. Quilliot. Scheduling with communication delays and data routing in message passing architectures. In *Parallel and Distributed Processing*, volume 1388 of *Lecture Notes in Computer Science*, pages 438–451, 1998.
- [11] L. Otten and R. Dechter. Towards parallel search for optimization in graphical models. In *Proc. of the 11th Int. Symposium on Artificial Intelligence and Mathematics*, 2010.
- [12] P. Padhy, R. K. Dash, K. Martinez, and N. R. Jennings. A utility-based sensing and communication model for a glacial sensor network. In *Proc. of 5th Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pages 1353–1360, 2006.
- [13] M. A. Paskin. *Exploiting locality in probabilistic inference*. PhD thesis, University of California at Berkeley, 2004.
- [14] M. A. Paskin and C. E. Guestrin. Robust probabilistic inference in distributed systems. In *Proceedings of the 20th Conf. on Uncertainty in Artificial Intelligence*, pages 436–445, 2004.
- [15] A. Petcu. *A Class of Algorithms for Distributed Constraint Optimization*. Phd. thesis no. 3942, Swiss Federal Institute of Technology (EPFL), 2007.
- [16] R. Stranders, A. Farinelli, A. Rogers, and N. Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence*, pages 292–298, 2009.
- [17] G. Tel. *Introduction to Distributed Algorithms*. Cambridge University Press, 2000.
- [18] H. Topcuoglu, S. Hariri, and M.-Y. Wu. Task scheduling algorithms for heterogeneous processors. *Proc. of Heterogeneous Computing Workshop*, pages 3–14, 1999.
- [19] Y. Xia and V. K. Prasanna. Parallel exact inference on the cell broadband engine processor. In *Proc. of the 2008 ACM/IEEE Conf. on Supercomputing*, pages 1–12, 2008.

Bounded Decentralised Coordination over Multiple Objectives

Francesco M. Delle Fave, Ruben Stranders, Alex Rogers & Nicholas R. Jennings
University Of Southampton
{fmdf08r,rs2,acr,nrj}@ecs.soton.ac.uk

ABSTRACT

We propose the bounded multi-objective max-sum algorithm (B-MOMS), the first decentralised coordination algorithm for multi-objective optimisation problems. B-MOMS extends the max-sum message-passing algorithm for decentralised coordination to compute bounded approximate solutions to multi-objective decentralised constraint optimisation problems (MO-DCOPs). Specifically, we prove the optimality of B-MOMS in acyclic constraint graphs, and derive problem dependent bounds on its approximation ratio when these graphs contain cycles. Furthermore, we empirically evaluate its performance on a multi-objective extension of the canonical graph colouring problem. In so doing, we demonstrate that, for the settings we consider, the approximation ratio never exceeds 2, and is typically less than 1.5 for less-constrained graphs. Moreover, the runtime required by B-MOMS on the problem instances we considered never exceeds 30 minutes, even for maximally constrained graphs with 100 agents. Thus, B-MOMS brings the problem of multi-objective optimisation well within the boundaries of the limited capabilities of embedded agents.

Categories and Subject Descriptors

I.2.11 [Distributed Artificial Intelligence]: Multiagent Systems

General Terms

Algorithms, Theory, Experimentation

Keywords

Coordination, Distributed Problem Solving

1. INTRODUCTION

Many real world problems involve the optimisation of multiple, possibly conflicting, objectives. Examples of bi-objective problems include the use of unmanned aerial vehicles (UAVs) for searching for survivors, while simultaneously establishing a wireless communication network between them [17], and controlling the motion of mobile robots to minimise travel distance while preventing collisions [14]. In both cases, independently maximising one objective results in detrimental

Cite as: Bounded Decentralised Coordination over Multiple Objectives, F. M. Delle Fave, R. Stranders, A. Rogers and N.R. Jennings, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Yolum, Tumer, Stone and Sonenberg (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 371-378.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

performance in terms of the other. Now, due to the potential life-critical nature of these problems, centralised control of these UAVs or ground robots is not desirable, since it creates a single point of failure. Consequently, the use of multi-agent technology has been advocated to achieve reliable, robust and scalable control within these sensitive scenarios [7, 9]. Specifically, many decentralised constraint optimisation algorithms have been proposed to allow multiple agents to coordinate their actions in an attempt to achieve their collective goals [4, 5, 10, 13, 15].

However, whilst both multi-objective constraint optimisation and decentralised coordination has generated considerable interest, no decentralised multi-objective optimisation algorithms have been proposed in previous work.

On the one hand, the field of decentralised optimisation has focused exclusively on problems involving a single objective, which are often represented as *decentralised constraint optimisation problems* (DCOPs). A number of algorithms have been proposed for solving general DCOPs, which can be divided in three broad classes. The first contains algorithms that are designed to find optimal solutions, such as ADOPT [13] and DPOP [15], but have a computational or communication complexity that is exponential in the number of agents. The second is composed of algorithms, such as the distributed stochastic algorithm (DSA) [5] or Maximum Gain Message [10], designed for large multi-agent systems, but which often converge to poor quality solutions. However, there exists a third class of algorithms usually referred to as Generalised Distributive Law (GDL) [1], which constitutes a compromise between the extremes represented by the first two classes, and can be used to compute good quality approximate solutions. In particular, one GDL algorithm, the max-sum algorithm, has been shown to generate solutions closer to the optimum than (for example) DSA, while being robust against message loss and exhibiting a scalable computational and communication cost [4]. An inherent shortcoming of the max-sum algorithm, however, is that it is not guaranteed to converge on cyclic constraint graphs, in which case it can perform arbitrarily poorly. This limits its applicability in safety-critical domains. The *bounded* max-sum algorithm, an extension to the standard max-sum algorithm, addresses this shortcoming, by pruning the constraint graph to a tree. In so doing, it is capable of providing performance guarantees on the computed solutions [3].

On the other hand, research on multi-objective constraint optimisation has yielded two extensions to well known single-objective algorithms, such as bucket elimination and branch and bound [16, 11], to solve general multi-objective optimisation problems. However, such approaches are centralised, and therefore lack the robustness required in the aforemen-

tioned scenarios. Moreover, since these algorithms exhibit a computation cost that is exponential in the number of agents, they are capable of solving only the smallest of problem instances. Another line of research has investigated extensions to standard DCOP algorithms for solving single-objective DCOPs with resource constraints [2], such as a capacity constraints in power distribution networks [8]. However, these resource constraints are not expressive enough to represent general multi-objective problems.

Thus, against this background, we identify a need for a decentralised coordination algorithm that has scalable computational and communication costs, and can provide good quality solutions for problems involving multiple objectives. To address this requirement, we propose the first decentralised coordination algorithm for multi-objective optimisation problems; the bounded multi-objective max-sum algorithm (B-MOMS). B-MOMS extends the bounded max-sum algorithm to solve multi-objective DCOPs (MO-DCOPs), a novel extension to the DCOP framework. These MO-DCOPs are encoded into a bipartite factor graph, in which vertices represent variables (owned by agents) and multi-objective functions, and edges represent the dependencies between the two. The B-MOMS algorithm then operates by exchanging messages between the variables and functions to compute approximate solutions to an MO-DCOP, whilst providing quality guarantees.

In more detail, this paper makes the following contributions to the state of the art:

- We propose the MO-DCOP problem, a general formalism for multi-objective coordination problems, which generalises the well known DCOP framework to the multi-objective setting.
- We develop B-MOMS, the first bounded decentralised algorithm for solving multi-objective optimisation problems. The operation of B-MOMS consists of three phases:
 - The first extends the bounded max-sum algorithm [3] to compute a cycle-free sub-graph of the multi-objective factor graph. To achieve this, we generalise the maximum spanning tree problem solved by the previous algorithm to handle the vector weights that we face in MO-DCOPs.
 - The second generalises the key mathematical operators required by max-sum to optimally solve the multi-objective problem encoded in this cycle-free factor graph.
 - Since there might be multiple Pareto optimal assignments to the cycle-free problem, the third and final phase enables agents to reach consensus on which global assignment to choose.
- We prove B-MOMS is optimal in acyclic factor graphs, and derive problem dependent approximation bounds in general graphs that do contain cycles.
- We present an extensive empirical evaluation of B-MOMS by benchmarking against a centralised optimal algorithm on a multi-objective extension of the graph colouring problem. We demonstrate that the approximation ratio never exceeds 2, even for extremely constrained problems (i.e. fully connected graphs), and

is less than 1.5 for graphs where constraints exists between 20% of all pairs of agents for 14 variables. Moreover, the results indicate that the runtime required by B-MOMS never exceeds 30 minutes, even for maximally constrained graphs with 100 agents, positioning it well within the confines of many real-life applications.

The remainder of this paper is organised as follows. In Section 2 we discuss the theoretical background of B-MOMS. In Section 3, we formalise the MO-DCOP framework. In Section 4, we develop the algorithm, describing each of its three phases, and describe its theoretical properties in Section 5. Finally, we empirically evaluate B-MOMS in Section 6. Section 7 concludes.

2. PRELIMINARIES

In this section we review the theoretical background of our algorithm. In particular, in Section 2.1, we introduce fundamental concepts related to multi-objective optimisation and in Sections 2.2 and 2.3 we discuss the max-sum algorithm and the bounded max-sum algorithm respectively.

2.1 Multi-Objective Optimisation

A multi-objective optimisation problem (MOOP) is defined as the problem of simultaneously maximising k incommensurable *objective functions*, defined over a set $\mathbf{x} = \{x_1, \dots, x_M\}$ of M discrete variables, where each x_j takes values in a discrete domain $D_{x_j} = \{d_j^1, \dots, d_j^{|D_{x_j}|}\}$. Thus, a solution to a MOOP is an assignment $\mathbf{a}^* = \{(x_1 = d_1^{(1)}), \dots, (x_M = d_M^{(M)})\}$ of values to variables, such that:

$$\mathbf{a}^* = \arg \max_{\mathbf{a} \in D_{\mathbf{x}}} \mathbf{U}(\mathbf{x}) = [U^1(\mathbf{x}), \dots, U^k(\mathbf{x})]^T \quad (1)$$

where $D_{\mathbf{x}} = \times_{j=1}^M D_{x_j}$ is the domain of variables \mathbf{x} . Here, each objective function U^i can be defined over a subset $\mathbf{x}_i \subseteq \mathbf{x}$ of the variables of the problem. However, for ease of exposition, we assume each function is defined over the same set of variables.

Now, since the functions are incommensurable, it is possible that multiple assignments satisfy Equation 1. For example, consider three assignments \mathbf{a}_1 , \mathbf{a}_2 and \mathbf{a}_3 , such that $\mathbf{U}(\mathbf{a}_1) = [4, 5]$, $\mathbf{U}(\mathbf{a}_2) = [4, 3]$, and $\mathbf{U}(\mathbf{a}_3) = [6, 3]$. Clearly we have that $[4, 5]$ and $[6, 3]$ are larger than $[4, 3]$. However, $[4, 5]$ and $[6, 3]$ are not comparable. Thus, \mathbf{a}_2 does not satisfy Equation 1. Indeed, Equation 1 involves the optimisation of sets of *partially-ordered* assignments. Thus, to characterise the optimal solutions of a multi-objective optimisation problem, we use the well known concept of *Pareto optimality*:

DEFINITION 1 (PARETO OPTIMALITY [12]). *An assignment $\mathbf{a}^* \in D_{\mathbf{x}}$ is Pareto optimal iff there does not exist another assignment $\mathbf{a} \in D_{\mathbf{x}}$, such that $\mathbf{U}(\mathbf{a}) \geq \mathbf{U}(\mathbf{a}^*)$, and $U^i(\mathbf{a}) > U^i(\mathbf{a}^*)$ for at least one objective function.*

The utility vector $\mathbf{U}(\mathbf{a}^*)$ corresponding to a Pareto optimal assignment \mathbf{a}^* is referred to as a *non-dominated* vector. We define the notion of *non-dominance* as follows:

DEFINITION 2 (NON-DOMINANCE). *A vector $\mathbf{U}(\mathbf{a}^*) \in D_{\mathbf{x}}$ is non-dominated iff there does not exist an assignment $\mathbf{a} \in D_{\mathbf{x}}$, such that $\mathbf{U}(\mathbf{a}) \geq \mathbf{U}(\mathbf{a}^*)$, with at least one $U^i(\mathbf{a}) > U^i(\mathbf{a}^*)$. Otherwise, $\mathbf{U}(\mathbf{a}^*)$ is said to be dominated.*

Thus, since a multi-objective problem involves the optimisation over partially ordered assignments, its solution is a *set* of Pareto optimal assignments. In the remainder of this paper, we will refer to the set of Pareto optimal assignments as **PO**.

2.2 The Max-Sum Algorithm

The max-sum algorithm is a decentralised message-passing optimisation algorithm belonging to the generalised distributive law (GDL) framework [1]. GDL algorithms exploit the factorisability of many optimisation problems, to solve them in an effective and efficient manner. Particularly, such problems are characterised as optimisation problems where the valuation algebra of the global constraint function is a *commutative semi-ring* (i.e. where the distributive law holds). All standard DCOP problems exhibit this characteristic [4].

Now, the most general characterisation of a DCOP involves M agents, each controlling a single discrete variable x_j , $j \in [1, M]$. Constraints between agents are encoded as functions $U_i(\mathbf{x}_i)$ ($i \in [1, N]$) over these variables. The scope $\mathbf{x}_i \subseteq \mathbf{x}$ of constraint function U_i contains the variables of the agents over which the constraint is defined. The aim of the coordination problem is then to choose variable assignments that maximise the sum of the constraint functions:

$$U(\mathbf{x}) = \sum_{i=1}^N U_i(\mathbf{x}_i) \quad (2)$$

In order to use the max-sum algorithm the problem is encoded as a special bipartite graph called a factor graph, in which vertices represent variables and functions, and edges the dependencies between them.

Max-sum defines two types of messages that are exchanged between variables and functions:

From variable x_j to function U_i :

$$q_{j \rightarrow i}(x_j) = \sum_{k \in M(j) \setminus i} r_{k \rightarrow j}(x_j) \quad (3)$$

where $M(j)$ represents the set of indices of the functions connected to variable x_j (i.e. the functions in which x_j occurs as an argument).

From function U_i to variable x_j :

$$r_{i \rightarrow j}(x_j) = \max_{\mathbf{x}_i \setminus x_j} \left(U_i(\mathbf{x}_i) + \sum_{k \in N(i) \setminus j} q_{k \rightarrow i}(x_k) \right) \quad (4)$$

where $N(i)$ represents the set of indices of the variables connected to function U_i .

Note that both $q_{j \rightarrow i}(x_j)$ and $r_{i \rightarrow j}(x_j)$ are scalar functions of variable x_j . When the factor graph is acyclic, these messages represent the maximum aggregate utility possible over the respective components of the graph formed by removing the dependency between U_i and x_j , for each value $d \in D_{x_j}$ in the domain of variable x_j . Thus, in this case, the marginal function of each variable is calculated by:

$$z_j(x_j) = \sum_{i \in M(j)} r_{i \rightarrow j}(x_j) = \arg \max_{\mathbf{x} \setminus x_j} \sum_{i=1}^N U_i(\mathbf{x}_i) \quad (5)$$

after which the optimal assignment of x_j is found by:

$$a_j = \arg \max_{x_j} z_j(x_j)$$

2.3 The Bounded Max-Sum Algorithm

When the factor graph is cyclic, the straightforward application of max-sum is not guaranteed to converge. However, by pruning edges such that an acyclic sub-graph of the factor graph is obtained, a bounded approximation can be derived [3]. More specifically, here, the goal is to compute a variable assignment $\tilde{\mathbf{a}}$ in the acyclic factor graph, such that:

$$V^* = \sum_{i=1}^N U_i(\mathbf{a}_i^*) \leq \rho \sum_{i=1}^N U_i(\tilde{\mathbf{a}}_i) = \rho \tilde{V}$$

where \mathbf{a}^* is the optimal solution of the cyclic factor graph, and ρ is the approximation ratio. To ensure this approximation ratio is as small as possible, the algorithm prunes those edges that have the least impact on solution quality. The impact of an edge between x_j and U_i is defined as its weight w_{ij} , which is computed by:

$$w_{ij} = \max_{\mathbf{x}_i \setminus x_j} \left[\max_{x_j} U_i(\mathbf{x}_i) - \min_{x_j} U_i(\mathbf{x}_i) \right] \quad (6)$$

Once all the weights are computed, the GHS algorithm [6] is used to compute a maximum spanning tree of the factor graph in a decentralised fashion. The newly obtained acyclic factor graph is then used in the second phase, in which the max-sum algorithm is used to compute $\tilde{\mathbf{a}}$, which is the optimal variable assignment to the modified problem:

$$\tilde{V}_m = \sum_i \min_{\mathbf{x}_i^c} U_i(\tilde{\mathbf{a}}_i)$$

where \mathbf{x}_i^c is the set of variables that were eliminated from the scope of function U_i , corresponding to the edges that were pruned from the factor graph.

The approximation ratio ρ is now given by:

$$\rho = 1 + (\tilde{V}_m + W - \tilde{V}) / \tilde{V} \quad (7)$$

where W is the sum of the weights of the pruned edges. Thus, an upper bound on the optimal solution can be computed as follows:

$$\tilde{V}_m + W \geq V^*$$

3. MO-DCOP FORMALISATION

We formalise the general problem by extending the DCOP framework to the setting of multiple objective functions. More formally, we consider the multi-objective DCOP (MO-DCOP) problem, which involves the simultaneous optimisation of k DCOPs, where each DCOP is defined as in Section 2.2. Specifically, we consider the problem of maximising the following vector of objective functions:

$$\mathbf{U}(\mathbf{x}) = \left[U^1(\mathbf{x}), \dots, U^k(\mathbf{x}) \right]^T \quad (8)$$

Since each component of $U(\mathbf{x})$ is a DCOP, this global objective functions is decomposable into N factors, each of which is a multi-objective constraint function \mathbf{U}_i :

$$\mathbf{U}(\mathbf{x}) = \sum_{i=1}^M \mathbf{U}_i(\mathbf{x}_i)$$

where, again, each $\mathbf{x}_i \subseteq \mathbf{x}$ is the subset of variables representing the scope of multi-objective constraint function \mathbf{U}_i , which are defined as:

$$\mathbf{U}_i(\mathbf{x}_i) = \left[U_i^1(\mathbf{x}_i), \dots, U_i^k(\mathbf{x}_i) \right]^T$$

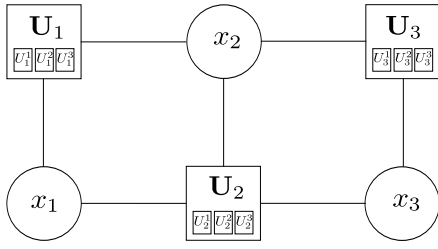


Figure 1: An example of a multi-objective factor graph, involving three variables, three objectives and three constraint functions.

Note that, for ease of exposition, we assume that all the local constraint functions U_i^k are defined over the same set of variables \mathbf{x}_i . However, this is not an essential requirement for the application of our algorithm.

Within this setting, the different solutions of a MO-DCOP are characterised using the solution concepts of Pareto optimality and non-dominance introduced in Section 2. Following these definitions, the solutions of a MO-DCOP are characterised as a set of Pareto optimal assignments PO corresponding to a set of non-dominated utilities vectors ND . Note that $\prod_{i=1}^M |D_i|$ is an upper bound of the number of possible solutions, which is equal to the size of the Cartesian product of the domains of all variables.

Finally, we encode the MO-DCOP as a multi-objective factor graph by representing each variable x_i of the MO-DCOP as a variable node, while each function node now represents a vector function \mathbf{U}_i . The following example illustrates such a multi-objective factor graph.

EXAMPLE 1. Consider the factor graph in Figure 1, with three variables x_1 , x_2 , and x_3 , which are controlled by agents \mathcal{A}_1 , \mathcal{A}_2 , and \mathcal{A}_3 . There are three constraints between the agents, represented by functions \mathbf{U}_1 , \mathbf{U}_2 , and \mathbf{U}_3 , each defined over three objectives U^1 , U^2 , and U^3 .

4. THE B-MOMS ALGORITHM

In this section we present the bounded multi-objective max-sum (B-MOMS) algorithm. B-MOMS extends the max-sum algorithm to compute solutions to MO-DCOPs. In detail, our algorithm proceeds in three phases:

- The *bounding (B)* phase, which extends the bounded max-sum algorithm discussed in Section 2.3, in order to provide quality guarantees. To achieve this, we generalise the maximum spanning tree algorithm to vector weights.
- The *max-sum (MS)* phase, during which the agents coordinate to find the Pareto optimal set of solutions to the cycle-free factor graph computed in the first phase. This requires a redefinition of the two key operations required by the max-sum algorithm (addition and marginal maximisation) for multiple objectives.
- The *value-propagation (VP)* phase, in which agents select a consistent variable assignment. This extends the standard *VP* phase [15] to the case where multiple non-commensurable alternatives exist.

In what follows, we discuss each phase in more detail.

4.1 The Bounding Phase

This phase builds upon the bounded max-sum algorithm described in Section 2.3 by extending the edge weights w_{ij} to the multi-objective vector weights. To this end, we first compute the impact of each variable x_j in the scope of each local multi-objective function \mathbf{U}_i over all k objectives:

$$\mathbf{w}_{ij} = [w_{ij}^1, \dots, w_{ij}^k]^T$$

Moreover, we define each scalar weight w_{ij}^o ($1 \leq o \leq k$) as in the single-objective case:

$$w_{ij}^o = \max_{\mathbf{x}_i \setminus x_j} \left[\max_{x_j} U_i^o(\mathbf{x}_i) - \min_{x_j} U_i^o(\mathbf{x}_i) \right]$$

Since the problem of finding a maximum spanning tree is defined on instances with scalar edge weights, it is necessary to rank the vector weights. This procedure must ensure that the resulting ordering favours deletion of dominated vectors over non-dominated ones. One way of doing this is to assign a scalar weight w_{ij} to each vector \mathbf{w}_{ij} , which is proportional to the number of edge weights it dominates. More formally:

$$w_{ij} = -|\{\mathbf{w}_{mn} \mid \mathbf{w}_{mn} > \mathbf{w}_{ij}, (i, j) \neq (m, n)\}|$$

Thus, using this scalarisation, non-dominated weight vectors are assigned a value of 0, vectors dominated by a single element are assigned a value of -1 , and so on. With these scalar edge weights, the GHS algorithm can be used to compute a maximum spanning tree as discussed in Section 2.3.

4.2 The Max-Sum Phase

The second phase executes max-sum on the acyclic factor graph resulting from the previous phase. In order to apply max-sum to the multi-objective setting, however, the messages exchanged between functions and variables (Equations 3 and 4) need to be generalised to vectors of constraint functions.

Recall from Section 2.2 that, if the factor graph is acyclic, the messages r and q exchanged between U_i and x_j represent the maximum aggregate utility over the two components of the graph obtained by removing the dependency between U_i and x_j . The same holds in the multi-objective case. However, instead of $q_{j \rightarrow i}(x_j)$ and $r_{i \rightarrow j}(x_j)$ being scalar functions of x_j , these messages now map the domain of x_j to a set of utility vectors. To see why this is true, note that in the multi-objective domain, maximum utility is now defined in terms of the dominance relation from Definition 2. Since this relation induces a partial ordering, more than one such maximum might exist. To illustrate this, consider the following example:

EXAMPLE 2. Suppose the following message $r_{i \rightarrow j}(x_j)$ is sent by function \mathbf{U}_i to variable x_j with domain $D_j = \{\text{Red}, \text{Green}, \text{Blue}\}$:

$$r_{i \rightarrow j}(x_j) = \begin{cases} \{[0, 0, 0]\} & \text{if } x_j = \text{Red} \\ \{[0, 1, 0]\} & \text{if } x_j = \text{Green} \\ \{[-1, 2, -1], [2, 1, -1]\} & \text{if } x_j = \text{Blue} \end{cases}$$

This message conveys the fact that, if x_j is assigned the value *Red*, the maximum possible utility obtained within the sub-graph connected to \mathbf{U}_i after removing the dependency on x_j is equal to $[0, 0, 0]$. Similarly, if $x_j = \text{Blue}$, there are two incomparable maxima (since neither dominates the other), namely $[-1, 2, -1]$ and $[2, 1, -1]$.

To compute these messages, the two key mathematical operators required by max-sum—the addition of two vector functions (Equations 3 and 4) and the marginal maximisation with respect to a single variable ($\max_{\mathbf{x}_i \setminus x_j}$ in Equation 4)—need to be defined for this domain. These operators were previously defined in the context of the multi-objective bucket elimination algorithm [16]. Here, however, we redefine these to compute the messages exchanged in the max-sum algorithm.

In more detail, to add two functions f and g defined over scope \mathbf{x} :

$$(f + g)(\mathbf{x}) = ND(\{\mathbf{v} + \mathbf{w} \mid \mathbf{v} \in f(\mathbf{x}); \mathbf{w} \in g(\mathbf{x})\})$$

where function $ND(A)$ filters out the dominated vectors from input set A . Using the $+$ operator we can compute the sum of the messages r (which are univariate functions of x_j) in Equation 3, as well as the addition of multi-variate function \mathbf{U}_j to the sum of univariate functions of *different* variables x_k in Equation 4.

The second operator, marginal maximisation ($\max_{\mathbf{x}_i \setminus x_j}$), takes as input a multi-variate vector function $f(\mathbf{x}_i)$ and outputs a function $f'(x_j)$:

$$f'(x_j = d) = ND \left(\bigcup_{\mathbf{d} \in D_{\mathbf{x}_i \setminus x_j}} f(\{\mathbf{x}_i \setminus x_j\} = \mathbf{d}, x_j = d) \right)$$

where $D_{\mathbf{x}_i \setminus x_j}$ is the Cartesian product of the domains of variables $\mathbf{x}_i \setminus x_j$.

At the end of the max-sum phase, each variable x_j computes its marginal function z_j (Equation 5) to obtain a set of Pareto optimal assignments A_j^* . Since there might be multiple optimal assignments, agents need to reach consensus on which global assignment to choose. Thus, in what follows, we define a value-propagation phase where the agents jointly choose a Pareto optimal solution among the ones computed by the max-sum phase.

4.3 The Value-Propagation Phase

The third and final phase, value-propagation, again operates on the cycle-free factor graph computed by the bounding approach. In this phase, variables and function nodes in this factor graph coordinate to select a consistent variable assignment.

Now, at the end of the *MS* phase, each variable computes the set A_j^* of marginal Pareto optimal variable assignments for x_j , which is obtained by maximising over the marginal function z_j :

$$A_j^* = \arg \max_{x_j} z_j(x_j)$$

If, for any variable x_j , $|A_j^*| > 1$, then there exists more than a single global optimal assignment in the acyclic factor graph: $|\mathbf{PO}| > 1$. If this is the case, a variable can select an assignment $a_j^* \in A_j^*$ at random, or one that satisfies some (logical) condition C . For example, a_j^* might be chosen such that objective 1 is maximised, subject to objective 2 being at least 4, or more formally, $a_j^* \in A_j^*$ and $\max z_j(a_j^*)_1$, subject to $z_j(a_j^*)_2 \geq 4$. To select an assignment that satisfies this condition, the value-propagation phase proceeds by passing messages between the variables and functions in the acyclic factor graph, and is thus fully decentralised. First, the variable x_r with the lowest index is chosen as the *root* of the tree, and is responsible for initiating the value propagation phase by selecting a Pareto optimal assignment a_r^*

that satisfies C . The variable then sends value-propagation messages ($x_r = a_r^*$) to all the function nodes to which the variable is connected.

The behaviour of all the other nodes in the graph will then depend on their type. More specifically:

Function nodes: Upon receiving a message ($x_j = a_j^*$) from variable x_j , multi-objective constraint function \mathbf{U}_i computes the set \mathbf{A}^* of local Pareto optimal assignments for variables $\mathbf{x}_i \setminus x_j$, conditioned on $x_j = a_j^*$:

$$\mathbf{A}^* = \left\{ \mathbf{a} \mid \mathbf{a} \in D_{\mathbf{x}_i \setminus x_j}, g(\mathbf{a}) \in ND \left(\bigcup_{\mathbf{a} \in D_{\mathbf{x}_i \setminus x_j}} g(\mathbf{a}) \right) \right\}$$

where $g(a)$ is defined as:

$$g(\mathbf{a}) = \mathbf{U}_i(\{\mathbf{x}_i \setminus x_j\} = \mathbf{a}, x_j = a_j) + \sum_{k \in N(i) \setminus j} q_{k \rightarrow i}(a_k)$$

After computing set \mathbf{A} , value propagation selects a Pareto optimal assignment $\mathbf{a}^* \in \mathbf{A}$ that satisfies condition C and sends the message ($x_k = a_k^*$) to every variable x_k ($k \neq j$), where a_k^* is the element of \mathbf{a}^* corresponding to x_k .

Variable nodes: For each non-root variable x_j , once it receives a message ($x_j = a_j^*$) from a function \mathbf{U}_i , it sets its value to a_j^* and propagates the message ($x_j = a_j^*$) to all the function nodes \mathbf{U}_k , $k \neq i$.

Note that, during value propagation, a single message is sent across each link in the factor graph. Thus, the algorithm terminates once each non-root variable has received a value-propagation message.

5. THEORETICAL ANALYSIS

We now discuss the theoretical properties of the B-MOMS algorithm.

5.1 Optimality of the MS Phase

The first property concerns the performance of the max-sum phase of the B-MOMS. Specifically, we show that the following theorem holds:

THEOREM 1. *The max-sum phase (MS) computes the entire set of Pareto optimal solutions \mathbf{PO} of the acyclic factor graph that is obtained by pruning edges during the bounding (B) phase of the algorithm.*

PROOF. The proof consists of two steps. Firstly, the valuation algebra defined by the $+$ and \max operators discussed in Section 4.2, together with the co-domain of the global multi-objective constraint function \mathbf{U} (Equation 8), is a commutative semi-ring [16]. Secondly, any GDL algorithm optimally solves problems whose valuation algebra is a commutative semi-ring, whenever the underlying constraint graph is acyclic [1]. Thus, since the second phase of B-MOMS is a GDL algorithm, the result holds. \square

Now, while solutions $\widetilde{\mathbf{PO}}$ are Pareto optimal in the acyclic sub-graph, they are not necessarily (or likely) Pareto optimal in the original factor graph. However, using Theorem 1, we can derive bounds on the quality of these solutions in the original factor graph.

5.2 Bounded Approximation

To derive these bounds, we follow a procedure similar to that of the bounded max-sum algorithm (Section 2.3). First, we define vector $\mathbf{W} = [W^1, \dots, W^k]$ as the sum of vector weights \mathbf{w}_{ij} of the edges between U_j and x_i that were pruned in the bounding phase to obtain an acyclic graph (Section 4.1). Furthermore, to characterise the upper bound computed by B-MOMS, we first define the concept of *utopia point*:

DEFINITION 3 (UTOPIA POINT [12]). *The utopia point \mathbf{V}^* of a multi-objective optimisation problem characterised by objective function $\mathbf{U}(\mathbf{x}) = [U^1(\mathbf{x}), \dots, U^k(\mathbf{x})]^T$ is given by:*

$$\mathbf{V}^* = \left[\max_{\mathbf{x}} U^1(\mathbf{x}), \dots, \max_{\mathbf{x}} U^k(\mathbf{x}) \right]$$

Put differently, the utopia point is the vector of values resulting from optimising each k DCOPs independently. Clearly, given any Pareto optimal assignment $\mathbf{a}^* \in \mathbf{PO}$, $\mathbf{U}(\mathbf{a}^*) \leq \mathbf{V}^*$ holds for any MO-DCOP. Thus, the utopia point is an upper bound of the value of a Pareto optimal solution of a MO-DCOP. Given these concepts we can state the following theorem:

THEOREM 2. *Given an arbitrary MO-DCOP, for any assignment $\tilde{\mathbf{a}} \in \overline{\mathbf{PO}}$ computed by B-MOMS, the following bound holds:*

$$\mathbf{U}(\tilde{\mathbf{a}}) + \mathbf{W} \geq \mathbf{V}^* \quad (9)$$

PROOF. The theorem follows directly by the fact that we extend the bounded max-sum algorithm for single objective DCOPs to the case of multi-objective problems. In more detail, for each objective o ($1 \leq o \leq k$) of an MO-DCOP, by using the approach defined in [3], it is easy to see that the following bound holds:

$$U^o(\tilde{\mathbf{a}}) + W^o \geq \max_{\mathbf{x}} U^o(\mathbf{x})$$

thus concluding the proof. \square

Similarly, we define the problem dependent approximation ratio $\boldsymbol{\rho} = [\rho_1, \dots, \rho_k]$ of the solutions computed by B-MOMS, where each ρ_i is given by Equation 7.

5.3 Complexity

Finally, we derive the computation and communication complexity of B-MOMS using properties inherited from the max-sum algorithm. Now, since B-MOMS exploits the factorisability of the problems it is solving, the scope of each constraint function $\mathbf{U}_i(\mathbf{x}_i)$ contains only the variables on which the constraint is defined. Therefore, computing message $r_{i \rightarrow j}$ from function \mathbf{U}_i to variable x_j (Equation 4) requires $O(|D_{max}|^{|\mathbf{x}_i|})$ evaluations of function \mathbf{U}_i , where D_{max} is the largest domain among variables \mathbf{x} . Hence, the computation is exponential *only* in the number of variables in the scope of \mathbf{U}_i , not the total number of variables.

Furthermore, since in the worst case every variable assignment of \mathbf{x} is Pareto optimal, after a certain (but finite) number have been exchanged, these messages contain $O(k \times |D_{max}|^{M+1})$ values: $|D_{max}|$ sets of vectors (one for each value in the variable's domain), each containing at most D_{max}^M non-dominated vectors of size k .

6. EMPIRICAL EVALUATION

Since B-MOMS is not an optimal algorithm, empirical evaluation is required to ascertain the quality of the solutions it computes. To this end, in this section we benchmark the performance of B-MOMS against an optimal algorithm. In the remainder of this section, we present a multi-objective extension to the canonical graph colouring problem used in our experiments, detail the experimental setup, and discuss the results.

6.1 Multi-Objective Graph Colouring

In order to evaluate the performance of our algorithm we consider a multi-objective extension of the graph-colouring problem, which is a well known benchmark problem in the DCOP literature [4]. In this multi-objective graph colouring problem, each agent \mathcal{A}_j owns a single variable x_j , taking values in the domain $D_j = \{Red, Green, Blue\}$. Within this setting, the agents' goals is to maximise the following multi-objective function:

$$\mathbf{U}(\mathbf{x}) = [U^1(\mathbf{x}), U^2(\mathbf{x}), U^3(\mathbf{x})]^T \quad (10)$$

where U^1, U^2, U^3 are the sum of bi-variate constraint functions $U_i^1(x_j, x_k), U_i^2(x_j, x_k), U_i^3(x_j, x_k)$ that exist among the variables \mathbf{x} . These three types of constraint functions are defined as follows:

Chromatic Difference: This objective function represents the common graph colouring conflict function:

$$U_i^1(x_j, x_k) = \begin{cases} 0 & x_j \neq x_k \\ -1 & x_j = x_k \end{cases} \quad (11)$$

Chromatic Ordering: This objective function imposes an ordering among the colours: *Red* = 1, *Green* = 2, and *Blue* = 3. Specifically, given two variables x_j and x_k where $j < k$, the variable with the higher index should have a higher ranked colour:

$$U_i^2(x_j, x_k) = \begin{cases} 0 & \text{if } j < k \text{ and } x_j < x_k \\ -1 & \text{otherwise} \end{cases} \quad (12)$$

Chromatic Distance: This objective is similar to the chromatic ordering. However, it considers the distance between the colours of different variables. In more detail, given two variables x_j and x_k , the distance of the colours between the two variable should equal one:

$$U_i^3(x_j, x_k) = \begin{cases} 0 & \text{if } |x_j - x_k| = 1 \\ -1 & \text{otherwise} \end{cases} \quad (13)$$

Thus, given an arbitrary graph $G = (V, E)$, we construct a factor graph as follows. Each vertex is represented as a variable x . Furthermore, for each edge (v, v') the factor graph contains a three-objective constraint function $\mathbf{U}_i(x_j, x_k) = [U_i^1(x_j, x_k), U_i^2(x_j, x_k), U_i^3(x_j, x_k)]^T$ where x_j and x_k are the variables corresponding to vertices v and v' .

6.2 Experimental Setup

We analyse the performance of B-MOMS by executing it on several instances of the multi-objective graph colouring problem defined previously. Specifically, we randomly generate *connected* graphs $G = (V, E)$ with a varying number $M = |V|$ of vertices, and varying graph density $\delta \in$

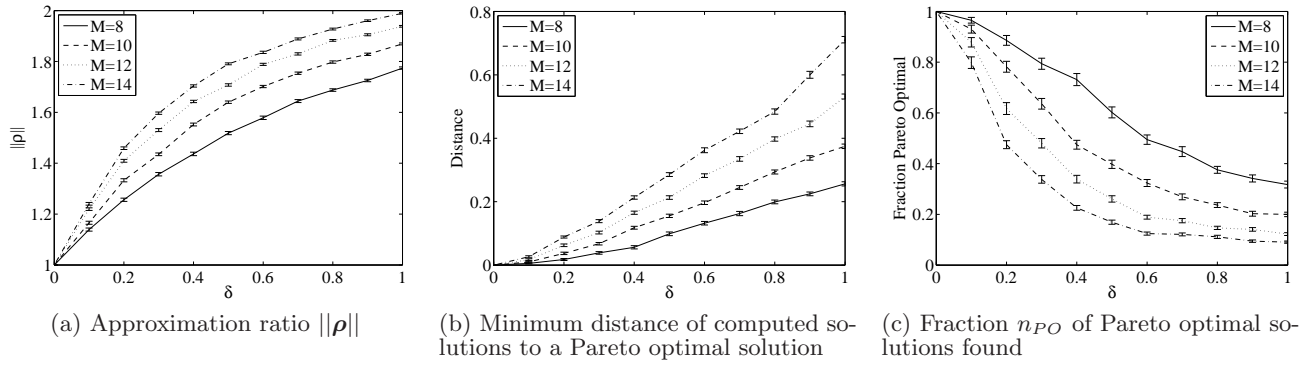


Figure 2: Empirical results for $M \leq 14$. Errorbars indicate the standard error of the mean.

$[0, 1]$. This latter parameter defines the constrainedness of the problem by controlling the number of edges; when $\delta = 0$ the number of edges $|E| = M - 1$, and the resulting graph G is a tree. Conversely, when $\delta = 1$, G is a complete graph, and $|E| = \frac{1}{2}M(M - 1)$.

We generated problem instances with $M = \{8, 10, 12, 14, 20, 40, 60, 80, 100\}$ and $\delta = \{0.0, 0.1, \dots, 1.0\}$. Moreover, for each combination of values for M and δ we ran B-MOMS 120 times to achieve statistical significance. We measure the performance of B-MOMS using the following four metrics:

1. The runtime (in milliseconds) required by B-MOMS to compute the set of solutions.
2. The average approximation ratio of the solutions $\widetilde{\mathbf{PO}}$ computed by B-MOMS. More specifically, we defined this as the norm of the approximation ratio vector $\|\rho\|$ defined in Section 5.2.
3. The minimum Euclidean distance between solutions $\mathbf{a} \in \widetilde{\mathbf{PO}}$ computed by B-MOMS and an optimal solution $\mathbf{a}^* \in \mathbf{PO}$. More formally, we calculate this distance as follows:

$$d(\mathbf{a}) = \left\| \min_{\mathbf{a}^* \in \mathbf{PO}} [\mathbf{U}(\mathbf{a}) - \mathbf{U}(\mathbf{a}^*)] \right\|$$

4. Finally, we measure the fraction Pareto optimal solutions found by B-MOMS:

$$n_{PO} = \frac{|\widetilde{\mathbf{PO}} \cap \mathbf{PO}|}{|\mathbf{PO}|}$$

Note that metrics 2 and 3 aggregate the objectives by using the notions of norm and Euclidean distance, which implies commensurability of the objectives. However, these metrics have been widely used in multi-objective literature [18]. Moreover, metrics 3 and 4 require the availability of the set of Pareto optimal solutions \mathbf{PO} . To compute these, we used a centralised brute-force optimal algorithm. This optimal algorithm exhaustively enumerates the Cartesian product of the domains of \mathbf{x} , and thus, has a computational complexity that is exponential in the number of variables. As a result, we do not report metrics 3 and 4 for $M > 14$, since we were unable to run a sufficient number of experiments to obtain statistically significant results. However, we do report metrics 1 and 2 for M up to 100.

6.3 Results and Discussion

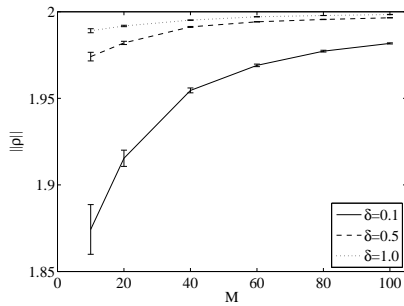
For $M \leq 14$, results are shown in Figure 2. First of all, all three plots confirm that the algorithm is optimal for acyclic graphs ($\delta = 0$), as proved by Theorem 1. Moreover, by increasing the number of constraints of the problem (i.e. by increasing δ), we can observe that the performance of B-MOMS degrades gracefully in terms of the approximation ratio, distance, as well as the fraction of optimal solutions found. Moreover, the approximation ratio never exceeds 2 (i.e. the value of the computed solution is greater than half of that of the optimal solution) even for extremely constrained problems, and is close to the value of 1.27 reported for the single-objective bounded max-sum algorithm [3] when each variable is involved in three constraints (corresponding to $\delta = 0.3$ for $M = 14$). Most importantly, for relatively sparse graphs ($\delta \approx 0.2$), which are often found in real-life multi-agent applications, B-MOMS recovers roughly 50% of the optimal solutions for $M = 14$.

Figures 3(a) and 3(b) report the approximation ratio and the runtime for larger problem instances. Specifically, Figure 3(a) clearly shows that, even for large instances, the approximation ratio again never exceeds 2, demonstrating the effectiveness of the bounding approach. Furthermore, 3(b) gives strong empirical evidence of the practical applicability of the algorithm. Despite the exponential relation between the number of variables and the time required by B-MOMS¹, for $M = 100$ and a maximally constrained problem, this time does not exceed 30 minutes. Moreover, it is important to note that these experiments were run on a single processor, while the computational load in a multi-agent system is shared among multiple computational entities. This brings B-MOMS well within the realm of the limited computational capacities of embedded agents found in many real-life applications.

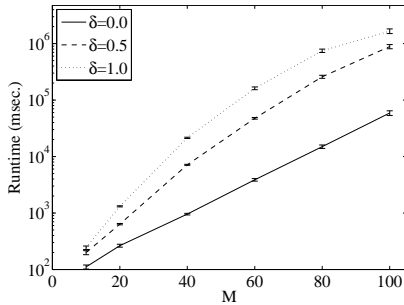
7. CONCLUSIONS

In this paper, we proposed the bounded multi-objective max-sum algorithm (B-MOMS), the first decentralised coordination algorithm for multi-objective optimisation problems. B-MOMS extends the bounded max-sum algorithm for decentralised coordination [3] to compute bounded approxi-

¹This is partially due to our implementation of the value-propagation phase, which for the purpose of these experiments, recovers an exponential number of (approximately) Pareto optimal solutions, instead of just a single one.



(a) The approximation ratio $\|\rho\|$ for varying number of vertices



(b) The runtime for varying number of vertices

Figure 3: Empirical results for $10 \leq M \leq 100$. Error bars indicate the standard error of the mean.

mate solutions to multi-objective DCOPs (MO-DCOPs), a novel extension to the DCOP framework. It consists of three phases. The first phase extends the bounded max-sum algorithm developed for max-sum [3] to compute a cycle-free sub-graph of the multi-objective factor graph, which involves a generalisation of the maximum spanning tree problem to vector weights. The second phase generalises the key mathematical operators required by max-sum to optimally solve the multi-objective problem encoded in this cycle-free factor graph. Since there might be multiple Pareto optimal assignments to the cycle-free problem, the third and final phase enables agents to reach consensus on which global assignment to choose. We proved the optimality of B-MOMS in acyclic constraint graphs, and derived bounds on the approximation ratio. Furthermore, benchmarked B-MOMS against an optimal centralised algorithm on a multi-objective extension of the graph colouring problem. We demonstrate that the approximation ratio never exceeds 2, and is typically less than 1.5 for graphs in which dependencies exist between 20% of all pairs of agents. Moreover, the runtime required by B-MOMS never exceeds 30 minutes, even for maximally constrained graphs with 100 agents, positioning it well within the confines of real-life applications.

For future work, we intend to apply our approach on challenging real-world problems, such as the coordination of multiple mobile sensors, and power distribution networks [8]. Moreover, we would like to extend B-MOMS to the setting where there exist uncertainty about the constraint functions. This is a non-trivial extension, since it requires the exchange of vectors of probability distributions, instead of scalars, and a further generalisation of the two key mathematical operators required by max-sum.

8. ACKNOWLEDGMENTS

The work reported in this paper was funded as part of the ALADDIN (Autonomous Learning Agents for Decentralised Data and Information Systems) Project and is jointly funded by a BAE Systems and EPSRC (Engineering and Physical Research Council) strategic partnership (EP/C548051/1), and also by the Systems Engineering for Autonomous Systems (SEAS) Defence Technology Centre established by the UK Ministry of Defence.

9. REFERENCES

- [1] S. M. Aji and R. J. McEliece. The Generalized Distributive Law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.
- [2] E. Bowring, M. Tambe, and M. Yokoo. Multiply-constrained distributed constraint optimization. In *Proc. of the 5th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 1413–1420, 2006.
- [3] A. Farinelli, A. Rogers, and N.R. Jennings. Bounded approximate decentralised coordination using the max-sum algorithm. In *IJCAI-09 Workshop on Distributed Constraint Reasoning*, pages 46–59, 2009.
- [4] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max sum algorithm. In *Proc. of the 7th Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pages 639–646, 2008.
- [5] S. Fitzpatrick and L. Meertens. Distributed coordination through anarchic optimization. In Victor Lesser, Charles L. Ortiz, Jr., and Milind Tambe, editors, *Distributed Sensor Networks*, chapter 11, pages 257–295. Kluwer Academic Publishers, 2003.
- [6] R.G. Gallager, P.A. Humblet, and P.M. Spira. A distributed algorithm for minimum-weight spanning trees. In *ACM Transactions on Programming Languages and Systems*, volume 5, pages 66–77, 1983.
- [7] N. R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.
- [8] A. Kumar, B. Faltings, and A. Petcu. Distributed constraint optimisation with structured resource constraints. In *Proc. of the 8th Int. Conf. on Autonomous Agents and Multi-Agent Systems*, pages 923–930, 2009.
- [9] V. Lesser, C. L. Ortiz, and M. Tambe. *Distributed Sensor Networks, A Multiagent Perspective*. Kluwer Academic Publishers, 2003.
- [10] R. J. Maheswaran, J. Pearce, and M. Tambe. A family of graphical-game-based algorithms for distributed constraint optimization problems. In *Coordination of Large-Scale Multiagent Systems*, pages 127–146. Springer-Verlag, Heidelberg Germany, 2005.
- [11] R. Marinescu. Exploiting problem decomposition in multi-objective constraint optimization. In *Proc. of the 15th Int. Conf. on Principles and Practice of Constraint Programming*, pages 592–607. 2009.
- [12] R.T. Marler and J.S. Arora. Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, 26:369–395, 2004.
- [13] P.J. Modi, W. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence Journal*, 161:149–180, 2005.
- [14] A. Mouaddib, M. Boussard, and M. Bouzid. Towards a formal framework for multi-objective multi-agent planning. In *Proc. of the 6th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 801–808, 2007.
- [15] A. Petcu and B. Faltings. DPOP: A scalable method for multiagent constraint optimization. *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence*, pages 266 – 271, 2005.
- [16] E. Rollón. *Multi-Objective Optimization in Graphical Models*. PhD thesis, Universitat Politècnica de Catalunya, 2008.
- [17] A. Savikumar and C. Keng-Yan Tan. UAV swarm coordination using cooperative control for establishing a wireless communications backbone. In *Proc. of the 9th Int. Conf. on Autonomous Agents and Multiagent Systems*, pages 1157–1164, 2010.
- [18] D. A. Van Veldhuizen and G.B. Lamont. Multi-objective evolutionary algorithm research: A history and analysis. Technical report, Dept. Elec Comp. Eng. Graduate School of Eng., 1998.

Communication-constrained DCOPs: Message approximation in GDL with function filtering

Marc Pujol-Gonzalez, Jesus Cerquides, Pedro Meseguer, J. A. Rodriguez-Aguilar
Artificial Intelligence Research Institute (IIIA)
Spanish Scientific Research Council (CSIC)
Campus de la UAB, Bellaterra, Spain
{mpujol, cerquide, pedro, jar}@iiia.csic.es

ABSTRACT

In this paper we focus on solving DCOPs in communication constrained scenarios. The GDL algorithm optimally solves DCOP problems, but requires the exchange of exponentially large messages which makes it impractical in such settings. Function filtering is a technique that alleviates this high communication requirement while maintaining optimality. Function filtering involves calculating approximations of the exact cost functions exchanged by GDL. In this work, we explore different ways to compute such approximations, providing a novel method that empirically achieves significant communication savings.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*intelligent agents, multiagent systems*

General Terms

Algorithms, Design

Keywords

Distributed constraint reasoning, DCOP, GDL, Filtering

1. INTRODUCTION

Distributed constraint optimization (DCOP) is a model for representing multi-agent systems in which agents cooperate to optimize a global objective. There are several complete DCOP algorithms that guarantee global optimality such as ADOPT [11], DPOP [13], and its generalization GDL [1, 15]. Since DCOPs are NP-Hard [11], solving them requires either an exponential number of linear size messages (ADOPT) or a linear number of exponentially large messages (DPOP, GDL). Nonetheless, some application domains are specially communication constrained. For instance, data transmission is severely limited in wireless sensor networks [17], and bandwidth is a scarce resource in peer-to-peer networks [6]. An approach in these domains is to drop optimality in favor of lower complexity, approximate algorithms with weaker guarantees [7, 9, 16]. As an alter-

Cite as: Communication-constrained DCOPs: Message approximation in GDL with function filtering, M. Pujol-Gonzalez *et al.*, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 379-386.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

native, we aim at reducing the communication costs while keeping optimality.

Function filtering [2, 3] is a technique that reduces the size of exchanged messages, and can be readily applied to GDL. Essentially, GDL with function filtering exchanges approximations of the messages that would be sent by GDL. As a consequence, the resulting algorithm's efficiency is highly dependent on the strategy used to compute these approximations. Intuitively, *better* strategies produce better approximations, leading to larger communication savings.

We first review state-of-the-art approximation methods [5, 14], fitting them in a common framework of bottom-up approximations. However, these methods are designed for the centralized case. Hence, their purpose is to reduce the overall computation, disregarding any communication costs.

Our main contribution in this paper is a novel class of message approximation techniques, the top-down approximations, that are specifically aimed at lowering such communication costs. Thereafter, we present two realizations of this new approach, namely brute-force decomposition and zero-tracking decomposition. Finally, we empirically evaluate the overall computation time and communication costs of these methods on several experiments. The results show that top-down approximations outperform bottom-up ones, always achieving larger communication savings. Further, zero-tracking decomposition remains competitive in computational effort with respect to state-of-the-art approximation methods [5, 14] while transmitting much less information.

This paper is structured as follows. Firstly, Section 2 introduces the DCOP model, and Section 3 presents the GDL algorithm with function filtering. Next, Section 4 describes the message approximation problem we aim to solve. Then, Section 5 introduces the bottom-up approximation framework, which represents current state-of-the-art approaches to message approximation. Next, Section 6 introduces our novel top-down approximation framework. Section 7 provides an empirical evaluation of both bottom-up and top-down approximation methods. Finally, Section 8 draws the most important conclusions of this work.

2. DCOP

Distributed Constraint Optimization Problems (DCOPs) involve a finite set of variables, each taking a value in a finite discrete domain. Variables are related by cost functions that specify the cost of assigning certain values to a subset of variables. Costs are positive real numbers plus 0 and ∞ . A DCOP is defined as a tuple (X, D, C, A, α) , where $X = \{x_1, \dots, x_n\}$ is a set of n variables; $D = \{D(x_1), \dots, D(x_n)\}$

is a set of finite discrete domains; $D(x_i)$ is the finite set of x_i 's possible values; C is a set of cost functions; each cost function $f_S \in C$ is defined on the ordered set $S \subseteq X$ (S is f_S 's scope, $|S|$ is f_S 's arity), specifies the cost of every combination of values of variables in S , namely $f_S : \prod_{x_j \in S} D(x_j) \mapsto \mathbb{R}^+$; A is a set of p agents; and $\alpha : X \rightarrow A$ maps each variable to some agent.

The objective of DCOP algorithms is to find the assignment of individual values to variables, such that the total (aggregated) cost over all cost functions is minimized. We make the common assumption that there are as many agents as variables, each agent controlling one variable, so from now on the terms variable and agent will be used interchangeably.

Next, we introduce some definitions of concepts and operations that will be used throughout the rest of this paper. A *tuple* t_S , with scope S , is an ordered set of values assigned to each variable in the subset $S \subseteq X$. The *cost* of a complete tuple t_X that assigns a value to each variable $x_i \in X$ is the addition of all individual cost functions evaluated on that particular tuple. Whenever a complete tuple t_X yields a cost lower than a user-specified threshold, we say that it is a *solution*. Further, a solution t_X is an *optimal solution* if its cost is the minimum of all possible solutions.

DEFINITION 1 (MIN-MARGINAL). *The projection $t_S[T]$ of a tuple t_S to $T \subset S$ is a new tuple t_T , removing the values assigned to variables not found in T . The min-marginal $f_S[T]$ of a cost function f_S onto $T \subset S$ is a new cost function f_T where each tuple t_T is assigned the minimum cost among all tuples t_S whose projection to T is t_T .*

$$\forall t_T \quad f_T(t_T) = \min_{t_S[T]=t_T} f_S(t_S).$$

DEFINITION 2 (COMBINATION). *The combination of two cost functions f_S and f_T , written $f_S \bowtie f_T$, is a new cost function f_U defined over their joint domain $U = S \cup T$, s.t.:*

$$\forall t_U \quad (f_S \bowtie f_T)(t_U) = f_S(t_U[S]) + f_T(t_U[T])$$

Combination is an associative and commutative operation.

DEFINITION 3 (LOWER BOUND). *A function f_T is a lower bound of f_S , noted $f_T \leq f_S$, iff $T \subseteq S$ and for all t_S tuples, the value by f_T of its projection $t_S[T]$ is lower than or equal to the value by f_S of t_S :*

$$\forall t_S \quad f_T(t_S[T]) \leq f_S(t_S).$$

Given two lower bounds $f_T, f_U \leq f_S$, we say that f_T is at least as good as f_U iff $f_U \leq f_T$.

3. GDL

Several algorithms can optimally solve DCOPs [11, 13]. In particular, we consider the GDL algorithm [1], following the Action-GDL description [15]. GDL works over a special structure named junction tree (JT) [8], also known as joint tree or cluster tree. JTs represent a decomposition of the DCOP objective function into an equivalent, partially ordered sequence of combinations and marginalizations that are computationally easier to perform. It is partially ordered because those operations at the same tree-level can be carried out in parallel. Further, GDL is easily applicable to distributed solving because, given a DCOP where each agent holds a different variable, a JT can be computed in distributed form [12]. Additionally, in the distributed case, JTs

also represent the communication links that are going to be exploited, and what information they are going to exchange. As a result, each node of the JT (also known as clique) represents an agent operating over a subset of problem's variables and cost functions, whereas edges represent communication links that nodes will use by exchanging marginalizations of their problem parts onto their shared variables.

In short, the full serial version of GDL for the all-vertices problem (also known as DCTE [2]) works as follows [1, 15]: it sends messages up and down the JT, two messages per edge of the JT. Each message contains a cost function that summarizes the effect of the JT part from which this message comes on the considered agent. After exchanging these messages, each agent contains enough information to optimally solve its subproblem. However, it may happen that two optimal solutions t_1 and t_2 (with the same cost) exist, so some agents would choose t_1 while others choose t_2 . This may lead to assigning two different values to the same variable. To prevent this, the JT root decides the optimal assignment and informs its children in the JT, which recursively inform their own children and so on. In this way, a coherent optimal solution is selected. For a detailed description of GDL's operation, consult [1, 15].

Limiting Message Size. A major drawback of GDL is that the size of exchanged messages is exponential with respect to the number of variables in the JT edges (s). A strategy to alleviate this fact is to relax GDL to an approximate form, such that the size of messages can not exceed $exp(r)$, where $r < s$. This approach was first introduced in [2], where it is named DMCTE(r). Since messages' sizes are now limited, DMCTE(r) does not necessarily compute the optimal solution anymore. Nevertheless, it provides an approximate solution, as well as an interval [lower bound, upper bound] limiting the optimum cost. In general, the larger the r -arity parameter, the better the solution quality, but the higher the communication cost. Hence, the algorithm can also be iterated by slowly increasing r until an acceptable (or optimal when the lower bound equals the upper bound) solution is found.

Function Filtering. Function filtering [2, 3] is a technique to further reduce messages' sizes. Remember that messages are formed by cost functions (f of arity s in GDL, f' of arity r in DMCTE(r)) represented as tuples t , and their associated cost $f(t)$. The idea is to avoid sending those tuples t that, when combined with other functions, will have a cost equal to or higher than the current upper bound.

To see how this can be done, imagine the following situation: let t be a tuple that should be sent from agent i to agent j , and let UB be an upper bound of the global cost. After t arrival, agent j may realize that all possible combinations of t with its own cost functions reach or exceed the UB , so t can not be part of an optimal solution. In this situation, actually sending t is useless. Hence, if i was able to detect such irrelevant tuples, it could avoid sending them altogether, further reducing its message size. In GDL, function f containing t should be added with g , a function formed by combining all cost functions of agent j plus all cost functions received by agent j except the one from agent i . In the DMCTE(r) approximation, function f' containing t is a lower bound of the exact function f . If agent i knows any lower bound $g' \leq g$ before sending f' , it can detect some tuples that exceed the UB cost, and therefore avoid sending them. Removing such useless t tuples is called *filtering* f'

GDL (DCTE)	DMCTE($r=1$)	DIMCTEf
		Iteration 1: DMCTE($r=1$)
		Iteration 2:
$CF(C_2 \rightarrow C_1): f_2[S_1] = g_1 = \frac{y}{a} \Big \frac{1}{b} \Big 2$	$CF(C_2 \rightarrow C_1): f_2[S_1] = g_1 = \frac{y}{a} \Big \frac{1}{b} \Big 2$	$CF(C_2 \rightarrow C_1): \overline{f_2[S_1]}^{g_3} = g_1 = \frac{y}{a} \Big \frac{1}{b} \Big 2$
$CF(C_3 \rightarrow C_1): f_3$	$CF(C_3 \rightarrow C_1): \begin{cases} \text{a unary lower bound of } f_3, \\ \text{i.e. } f_3[y] = g_2 = \frac{y}{a} \Big \frac{3}{b} \Big 1 \end{cases}$	$CF(C_3 \rightarrow C_1): \overline{f_3}^{g_4} = \frac{y}{a} \Big \frac{3}{b} \Big \frac{3}{1}$ $\left[\begin{array}{l} \frac{y}{a} \Big \frac{3}{b} \Big \frac{3}{1} \\ \frac{a}{a} \Big \frac{3+1}{b} \Big \frac{3}{1} \\ \frac{a}{a} \Big \frac{5+1}{b} \Big \frac{3}{1} \geq UB \\ \frac{b}{b} \Big \frac{5+4}{b} \Big \frac{3}{1} \geq UB \\ \frac{b}{b} \Big \frac{1+4}{b} \Big \frac{3}{1} \end{array} \right]$
$CF(C_1 \rightarrow C_2): (f_1 \bowtie f_3)[S_1] = \frac{y}{a} \Big \frac{5}{b} \Big 3$	$CF(C_1 \rightarrow C_2): (f_1 \bowtie g_2)[S_1] = g_3 = \frac{y}{a} \Big \frac{3}{b} \Big 3$	$CF(C_1 \rightarrow C_2): \overline{(f_1 \bowtie f_3)[S_1]}^{g_1} = \frac{y}{a} \Big \frac{3}{b} \Big \frac{3}{3}$ $\left[\begin{array}{l} \frac{y}{a} \Big \frac{5}{b} \Big \frac{3}{3} \\ \frac{a}{a} \Big \frac{5+1}{b} \Big \frac{3+2}{3} \end{array} \right]$
$CF(C_1 \rightarrow C_3): f_1 \bowtie g_1 = \frac{y}{a} \Big \frac{1}{b} \Big 4$ $\left[\begin{array}{l} \frac{y}{a} \Big \frac{1}{b} \Big 4 \\ \frac{a}{a} \Big \frac{1}{b} \Big 4 \\ \frac{b}{b} \Big \frac{1}{b} \Big 4 \end{array} \right]$	$CF(C_1 \rightarrow C_3): \begin{cases} \text{a unary lower bound of } f_1 \bowtie g_1, \\ \text{i.e. } (f_1 \bowtie g_1)[y] = g_4 = \frac{y}{a} \Big \frac{1}{b} \Big 4 \end{cases}$	$CF(C_1 \rightarrow C_3): \overline{f_1 \bowtie g_1}^{g_2} = \frac{y}{a} \Big \frac{1}{b} \Big \frac{4}{4}$ $\left[\begin{array}{l} \frac{y}{a} \Big \frac{1}{b} \Big \frac{4}{4} \\ \frac{a}{a} \Big \frac{4+3}{b} \Big \frac{4}{4} \geq UB \\ \frac{a}{a} \Big \frac{1+3}{b} \Big \frac{4}{4} \\ \frac{b}{b} \Big \frac{7+1}{b} \Big \frac{4}{4} \geq UB \\ \frac{b}{b} \Big \frac{4+1}{b} \Big \frac{4}{4} \end{array} \right]$
$SS(C_1 \rightarrow C_2): \{y = b\}$	$SS(C_1 \rightarrow C_2): \{y = a\}$	$SS(C_1 \rightarrow C_2): \{y = b\}$
$SS(C_1 \rightarrow C_3): \{y = b, z = b\}$	$SS(C_1 \rightarrow C_3): \{y = a, z = b\}$	$SS(C_1 \rightarrow C_3): \{y = b, z = b\}$
	$BB(C_2 \rightarrow C_1): lb = 4, pub = 1$	$BB(C_2 \rightarrow C_1): lb = 5, pub = 2$
	$BB(C_3 \rightarrow C_1): lb = 4, pub = 5$	$BB(C_2 \rightarrow C_1): lb = 5, pub = 1$
	$BB(C_1 \rightarrow C_2): lb = 4, pub = 5$	$BB(C_1 \rightarrow C_2): lb = 5, pub = 3$
	$BB(C_1 \rightarrow C_3): lb = 4, pub = 1$	$BB(C_1 \rightarrow C_3): lb = 5, pub = 4$

Figure 1: Sequence of messages exchanged by GDL, DMCTE($r=1$) and DIMCTEf solving the instance of Figure 2. Messages in the same cell are exchanged in parallel.

with g' , noted $\overline{f'}^{g'}$.

Finally, notice that if we iterate DMCTE(r) with increasing r values, the function g' sent from agent j to agent i in the previous iteration is a lower bound of g . Hence, agent i can readily use g' to filter f' before sending it at the current iteration, leading to a lower size message. The resulting algorithm is known as DIMCTEf, consisting of three phases for each iteration: (1) Cost propagation: agents exchange approximate cost functions (CF messages), using cost functions of the previous iteration to filter cost functions at the current iteration. (2) Solution propagation: values for the variables in the separators of the JT are decided in a top-down manner (SS messages). (3) Bound propagation: agents exchange global lower bounds and local upper bounds (BB messages) among nodes of the JT, following the same communication strategy as cost propagation. For a detailed description of each phase and the structure of the different message types, the reader should consult [2].

Example. The toy example of Figure 2 allows us to illustrate the behavior of the above mentioned algorithms. Figure 1 shows messages exchanged by GDL (left), DMCTE($r=1$) (middle) and DIMCTEf (right). GDL performs exact solving. CF messages contain cost functions, while SS messages contain assignments of variables propagated top-down in the JT. After receiving them, each agent is able to compute the optimum cost (5) and the same global solution ($xyz \leftarrow bbb$). DMCTE($r=1$) performs approximate solving, where only cost functions of arity 1 ($r=1$) can be exchanged (when sending/receiving cost functions to/from C_3 , since initial cost functions are binary they are approximated by unary lower bounds). BB messages contain a lower bound lb and a partial upper bound pub of the cost of the solution propagated by SS messages. After receiving them, each agent is able to compute a lower bound (4) of the optimum cost, a global upper bound (6) and the

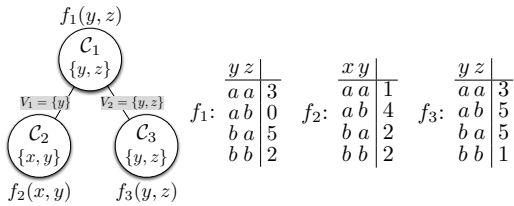


Figure 2: Toy example and a simple junction tree.

same solution of that cost ($xyz \leftarrow aab$). DIMCTEf performs exact solving with function filtering. The first iteration is DMCTE($r=1$), only unary cost functions can be sent. The second iteration, when also binary cost functions can be sent, uses cost functions of the first iteration (g_1, g_2, g_3, g_4) to filter current cost functions (the filtering process is shown between brackets). As result, some tuples are removed because they reach or exceed the UB computed at the previous iteration (6). Since the allowed arity equals the largest separator size, at the end of this iteration each agent is able to compute the minimum cost (5) and the same optimal solution ($xyz \leftarrow bbb$).

4. MESSAGE APPROXIMATIONS

During the cost propagation phase, GDL's task is to propagate cost functions in the form of min-marginals. DMCTE(r) relaxes this phase by sending lower-bound approximations instead of exact min-marginals, so that less information needs to be sent. Therefore, the more accurate these approximations are, the better results DMCTE(r) is going to achieve at the same iteration. Further, because greater accuracy means that function filtering will be able to prune more tuples, increasing the accuracy should also lower the total amount of communication needed to solve the problem.

Since we are interested in communication-constrained scenarios, from now on the bound r means that agents can not send functions of more than r variables. However, agents can compute functions of any arity. Consider an agent operating in DMCTE(r). Eventually, it will receive messages from all its children in the JT. Then, the agent combines this information with its own and marginalizes it onto the variables in the separator, to send the result to its parent. Nevertheless, since the agent is now constrained by the arity limit r , it can not send the exact min-marginal and it has to compute an approximation. Hence, the objective of the approximation task is to find a good lower bound for the min-marginal while communicating only functions of at most r variables. Some algorithms for this task have already been proposed in the literature [5, 14]. In the next section we review them and we fit them into a common framework which we call bottom-up approximation. In order to do that precisely, we need to introduce some additional definitions.

In the following, let $F = \{f_{T_1}, \dots, f_{T_n}\}$ be a set of functions, V a set of variables and r an arity limit.

$F = \{f_{xt}, f_{yt}, f_{zt}\}$	$f_{xt}: \begin{array}{c c} xt & \\ \hline aa & 2 \\ ab & 1 \\ ba & 3 \\ bb & 2 \end{array}$	$f_{yt}: \begin{array}{c c} yt & \\ \hline aa & 4 \\ ab & 1 \\ ba & 2 \\ bb & 2 \end{array}$	$f_{zt}: \begin{array}{c c} zt & \\ \hline aa & 0 \\ ab & 2 \\ ba & 0 \\ bb & 1 \end{array}$
$V = \{x, y, z\}$			
$r = 2$			

Figure 3: Example of functions to approximate.

We define $\bowtie F$, the combination of F , to be the function resulting from the joint combination of every function in F ,

$$\bowtie F = f_{T_1} \bowtie \dots \bowtie f_{T_n}.$$

The *min-marginal of F onto V* is the min-marginal of the combination of all the functions in F , that is $(\bowtie F)[V]$. In contrast, the *one-to-one min-marginal of F onto V* is the set containing the min-marginal of each of its functions f_{T_i} onto $T_i \cap V$, namely $F \downarrow V = \{f_{T_1}[T_1 \cap V], \dots, f_{T_n}[T_n \cap V]\}$.

Given a function f_V , we say that F is a *V -lower bound of f_V* iff the combination of the one-to-one min-marginal of F onto V is a lower bound of f_V , that is if $\bowtie(F \downarrow V) \leq f_V$.

Furthermore, F is an *(r, V) -lower bound of f_V* iff F is a V -lower bound of f_V and every function in $F \downarrow V$ has arity smaller than or equal to r .

Given F and G (r, V)-lower bounds, F is at least as good as G ($G \leq F$) iff $\bowtie(F \downarrow V)$ is at least as good as $\bowtie(G \downarrow V)$.

OBSERVATION 1. F is a V -lower bound of the min-marginal of F onto V . Formally,

$$\bowtie(F \downarrow V) \leq (\bowtie F)[V].$$

For any $f_S, f_T \in F$, the combination of f_S and f_T in F , is the set of functions that results from removing f_S and f_T from F and adding $f_S \bowtie f_T$, namely

$$F_{f_S \bowtie f_T} = (F \setminus \{f_S, f_T\}) \cup \{f_S \bowtie f_T\}.$$

Two functions f_S and f_T are *(r, V) -combinable* iff the min-marginal onto V of the combination of f_S and f_T can be expressed by a function of arity smaller than or equal to r . Any f_S and f_T such that $|(S \cup T) \cap V| \leq r$ are (r, V) -combinable.

The approximation task receives as input a set of functions F^0 , a set of variables V , and an arity limit r . Its goal is to find an (r, V) -lower bound for the min-marginal of F onto V . Since for a given approximation task the set of variables V is fixed, in the following we talk of r -combinable, r -lower bound, and min-marginal without explicitly mentioning V .

5. BOTTOM-UP APPROXIMATIONS

Informally, the fundamental idea behind current algorithms for min-marginal approximation is the following. If we combine any pair of functions from a set that is an r -lower bound of the exact min-marginal, the result is another lower bound which is at least as good as the original (and, in fact, most of the times better). Furthermore, if the two functions selected are r -combinable, then the result is also an r -lower bound.

The pseudocode for bottom-up approximation appears in Algorithm 1. Since by Observation 1 we know that F is a lower bound of the min-marginal of F onto V , a bottom-up algorithm starts from the original set of functions F^0 . At each iteration, the algorithm: (1) selects a pair of r -combinable functions (f_S, f_T) from the current set of functions; and (2) updates the set of functions to the combination of f_S and f_T in F , that is $F_{f_S \bowtie f_T}$. Since $F_{f_S \bowtie f_T}$ is also an r -arity lower bound and it is at least as good as F , the iterations are likely to improve the lower bound. When

Algorithm 1 Bottom-up approximation(F, V, r)

- 1: $(found, (f_S, f_T)) \leftarrow bestCombinablePair(F, V, r)$
 - 2: **while** $found$ **do**
 - 3: $F \leftarrow F_{f_S \bowtie f_T}$
 - 4: $(found, (f_S, f_T)) \leftarrow bestCombinablePair(F, V, r)$
 - 5: **end while**
 - 6: **return** $F \downarrow V$
-

no more pairs of r -combinable functions are found, the algorithm returns approximation represented by the last F .

5.1 Scope-based partitioning

Scope-based partitioning (SCP) is the most common bottom-up method [5]. Basically, it tries to combine as many functions as possible by choosing the two highest arity functions at each iteration, so long as they are r -combinable.

More in detail, the r -combinable pairs are selected as follows. First, the set of functions F is sorted decreasingly by arity and each function in the list is marked as non-finished. At each iteration, SCP takes the first non-finished element f_{S_1} of F and the element f_{S_i} of F closer to the head such that f_{S_1} and f_{S_i} are r -combinable. It removes them from F and inserts its combination at the head of the list. When there is no function f_{S_i} r -combinable with f_{S_1} , it marks f_{S_1} as finished. The algorithm proceeds until all functions are marked as finished.

Figure 4a depicts how SCP would compute an approximation for the example in Figure 3. Since all functions in F have the same arity (two), they are readily sorted. Hence, SCP would merge the two leftmost ones, and send the third one independently, resulting in the approximation:

$$F' = \{(f_{xt} \bowtie f_{yt})[xy], f_{zt}[z]\} = \left\{ \begin{array}{c|c} xy & \\ \hline aa & 2 \\ ab & 3 \\ ba & 3 \\ bb & 4 \end{array}, \begin{array}{c|c} z & \\ \hline a & 0 \\ b & 0 \end{array} \right\}$$

The main advantage of SCP is the low computational complexity that results from its simplicity. The algorithm performs a nested scanning through the list of functions. During this scanning process, the algorithm computes up to $|F| - 1$ function combinations. To determine the maximum arity of these functions, consider that S is the joint domain of all functions in F . Then, the number of variables that do not appear in V is $|S \setminus V|$. Since the arity limit is r , the maximum arity of each single merged function is $|S \setminus V| + r$. As a result, the complexity of the algorithm is $O(|F| \exp(|S \setminus V| + r))$.

5.2 Content-based partitioning

A major advantage of scope-based partitioning is its small computational overhead. Nonetheless, its main drawback is that it does not consider the information within each function. For instance, consider again the previous example. We showed that scope-based partitioning would produce partition F' above. However, notice that there are further approximations that satisfy the r -bound.

Content-based partitioning techniques guide the bottom-up approximation by consulting the functions' contents in addition to their scopes. In general, content-based partitioning tries to assess which pair of r -combinable functions yield the highest improvement.

In [14] Rollon and Dechter present a framework for content-based partitioning that implements the general approach

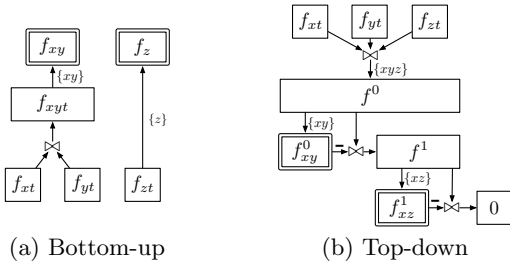


Figure 4: Examples of approximation strategies. Functions in a double-lined box are the ones finally sent.

outlined in Algorithm 1. Given an r -lower bound F , content-based partitioning provides the mechanism for selecting the best r -combinable pair of functions $f_S, f_T \in F$ such that the approximation represented by $F_{f_S \bowtie f_T}$ is better than the approximation represented by F . At each iteration, the technique: (1) generates every pair of r -combinable functions $f_S, f_T \in F$; (2) measures the gain obtained by combining f_S and f_T ; and (3) selects the pair that maximizes the gain.

Notice that the advantage of combining two functions before sending them is that they will be marginalized together. Hence, the gain can be calculated based on the difference between marginalizing together or marginalizing separately, which can be computed as:

$$f_V = (f_S \bowtie f_T)[V] - (f_S[V] \bowtie f_T[V]).$$

Therefore, the gain function is a metric that takes f_V as its input. Rollon and Dechter present two such functions. Firstly, the local relative error (LRE) metric, which is equivalent to the averaged 1-norm of f_V , assigns as gain the result of adding the costs of all tuples in f_V and dividing by the total number of tuples. Secondly, the local maximum relative error (LMRE) metric, which is equivalent to the ∞ -norm of f_V , assigns as gain the maximum cost of any tuple in f_V .

The downside of content-based decomposition is that, in the worst case, the algorithm performs up to $|F| - 1$ selections, computing $|F| - 1$ differences for each selection. Hence, the complexity of the algorithm is $O(|F|^2 \exp(|S \setminus V| + r))$.

6. TOP-DOWN APPROXIMATIONS

Bottom-up approximation methods focus on lowering the computational cost. Instead, our purpose is to primarily reduce communication costs. With this aim, we propose a new approach to generate approximations based on: (1) initially computing the function to approximate, and; (2) subsequently decomposing it into lower arity output functions. Figure 4b represents the process of building a top-down approximation of the example in Figure 3. As a first step, f_{xt} , f_{yt} and f_{zt} are combined and then marginalized onto $\{x, y, z\}$ to produce f^0 , the function to approximate. After that, the decomposition process starts. Firstly, consider that we select $S' = \{x, y\}$ out of all possible subsets of $\{x, y, z\}$ with arity two. Secondly, f^0 is marginalized onto S' to produce f_{xy}^0 , and this marginal is subtracted from f^0 to obtain f^1 (namely $f^1 = f^0 \bowtie (-f_{xy}^0)$).¹ Therefore, f^0 is decomposed as $f_{xy}^0 \bowtie f^1$, where f_{xy}^0 can be regarded as a function ready to communicate and f^1 as the *remainder*

¹Given f , we define $-f$ as f but changing the sign of f costs. It is easy to see that $f \bowtie -f$ is the null function.

Algorithm 2 Top-Down Approximation(F, V, r).

- 1: $f \leftarrow (\bowtie F)[V]$
 - 2: $F' \leftarrow \emptyset$
 - 3: $(found, f_{S'}) \leftarrow \text{selectBestMarginal}(f, r)$
 - 4: **while** $found$ **do**
 - 5: $F' \leftarrow F' \cup \{f_{S'}\}$
 - 6: $f \leftarrow f \bowtie (-f_{S'})$
 - 7: $(found, f_{S'}) \leftarrow \text{selectBestMarginal}(f, r)$
 - 8: **end while**
 - 9: **return** F'
-

after communicating f_{xy}^0 . The process continues searching for a decomposition for this remainder.

The main advantage of top-down over bottom-up methods is that they can represent a much wider space of approximations, and hence they should yield more accurate results. Unlike bottom-up methods, which start from an initial set of input functions and proceed by deciding which functions to join, top-down methods start from the function to approximate and proceed by successively selecting the best lower arity min-marginal. While such min-marginal is already part of the decomposition, the process continues by decomposing the result of subtracting the min-marginal from the function to approximate. In general a top-down approximation method is an iterative procedure that at each step i focuses on finding the *most informative min-marginal* for f^i whose arity is smaller than or equal to r . It incorporates the selected min-marginal, $f^{i-1}[S'_i]$, to the list of output functions and updates the function to approximate as follows:

$$f^i = f^{i-1} \bowtie (-f^{i-1}[S'_i]). \quad (1)$$

When the iterative process terminates, the following set of functions stands for the resulting decomposition of f :

$$F = \{f^0[S'_1], f^1[S'_2], \dots, f^n[S'_{n+1}]\}.$$

More in detail, a general top-down approximation method works as outlined in Algorithm 2. First, it computes the function to approximate (f) by combining the input functions in F and marginalizing onto V . After that, it uses some heuristic to select the best min-marginal $f_{S'}$. Finally, $f_{S'}$ is added to the set of output functions and subtracted from f . This process is repeated until no min-marginal provides additional information. In the remaining of the section we introduce two top-down approximation methods that implement the general method outlined in Algorithm 2.

6.1 Brute force decomposition

In order to determine the most informative min-marginal, a first approach is to consider every possible min-marginal onto r variables from V .² Then, we can readily use the gain functions from content-based partitioning to rank the min-marginals and select the most informative one.

This procedure has, however, a high computational cost. At the first iteration, it must compute $\binom{|V|}{r}$ marginals and evaluate them, each requiring $\exp(|V|)$ operations. At each following iteration, the number of marginals to compute decreases by one (the selected marginal is never computed again, but all others have to be reevaluated because f^i is

²Note that discarding functions whose arity is lower than r does not reduce the space of representable functions.

$i = 0$	aa	ab	ba	bb	C
xy					4
xz					4
yz					4

$i = 1$	aa	ab	ba	bb	C
xy	X				3
xz		X			3
yz		X			3

$i = 2$	aa	ab	ba	bb	C
xy	X	X			2
xz	X	X			2
yz	X	X	X	X	0

$i = 3$	aa	ab	ba	bb	C
xy	X	X	X	X	0
xz	X	X	X	X	0
yz	X	X	X	X	0

(a) Zeroes tracking table and counter vector

xyz	f^0	f^1	f^2	f^3
aaa	4	2	0	0
aab	2	0	0	0
aba	4	2	0	0
abb	3	1	0	0
baa	5	3	1	0
bab	3	1	1	0
bba	5	3	1	0
bbb	4	2	1	0

(b) Per iteration remainders

$f^0[\emptyset] = 2$
$f^1[yz] = \begin{array}{l} yz \\ aa \\ ab \\ ba \\ bb \end{array} \begin{array}{l} 2 \\ 0 \\ 0 \\ 2 \\ 1 \end{array}$
$f^2[xz] = \begin{array}{l} xz \\ aa \\ ab \\ ba \\ bb \end{array} \begin{array}{l} 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{array}$

(c) Selected min-marginals

Figure 5: Zero-tracking decomposition example.

different from f^{i-1}). Hence, its worst time complexity is $O\left(\binom{|V|}{r}^2 \cdot \exp(|V|)\right)$.

6.2 Zero-tracking decomposition

The main disadvantage of brute force decomposition is its high computational cost. Here we introduce zero-tracking decomposition, a top-down approximation method that aims at dodging this burden to reduce the computational cost.

Zero-tracking decomposition uses the zero norm of a min-marginal as the heuristic to assess its quality. The zero norm of a function is simply the number of elements in the domain whose image is not zero. Intuitively, if a function is only composed of zeros, it communicates no information whatsoever. The larger the number of non-zero entries in a function, the more informational it will be considered.

The reduction in computational cost comes from realizing that, at each iteration, there is a way to compute the zero norms of each min-marginal from the results of the previous iterations. This avoids the need for recomputing every min-marginal at each iteration as we pursue.

In what follows we detail the operation of the zero-tracking methods when applied to the example in Figure 3. Let $p = \binom{|V|}{r}$ be the number of possible r -arity min-marginals and q the number of tuples for each min-marginal.³ First, the algorithm allocates a boolean table *Zeroes* of p rows and q columns. Each entry $[U, t_U]$ in *Zeroes* encodes whether the value for t_U of the min-marginal of f onto U is zero or not. That is, $Zeroes[U, t_U]$ is true whenever $f[U](t_U)$ is zero. Hence, all entries are initialized to *false*. Additionally, it allocates a vector C of p integers to count the number of non-zero tuples for each min-marginal. Since C counts non-zero elements, it is initialized to the number of tuples in each min-marginal (q). At the top of Figure 5a we show (for iteration $i=0$) table *Zeroes* and vector C after initialization. Next, the exact min-marginal $(\bowtie F)[V]$ is calculated by combining all the initial functions and marginalizing the result onto $\{x, y, z\}$. The result is shown in Figure 5b as f^0 .

Notice that, at this point, f^0 does not contain any zero. In

³For simplicity of exposition we assume that all variables are defined over the same domain.

Algorithm 3 ZeroDecomposition(F, V, r).

```

1: initialize(Zeroes,  $C$ )
2:  $f \leftarrow (\bowtie F)[V]$ 
3:  $F' \leftarrow \emptyset$ 
4:  $(f_{S'}, \text{gain}) \leftarrow (f[\emptyset], 1)$ 
5: while  $\text{gain} > 0$  do
6:    $F' \leftarrow F' \cup \{f_{S'}\}$ 
7:    $f \leftarrow f \bowtie (-f_{S'})$ 
8:    $(f_{S'}, \text{gain}) \leftarrow \text{selectBestMarginal}(f, \textit{Zeroes}, C)$ 
9: end while
10: return  $F'$ 
11:
12: function selectBestMarginal( $f, \textit{Zeroes}, C$ )
13:   for all new  $t_S$  s.t.  $f(t_S) = 0$  do // new zeroes in  $f$ 
14:     for all  $U \in T$  do // subsets of  $r$  variables
15:       if not  $\textit{Zeroes}(U, t_S[U])$  then
16:          $C(U) \leftarrow C(U) - 1$ 
17:          $\textit{Zeroes}(U, t_S[U]) \leftarrow \textit{true}$ 
18:       end if
19:     end for
20:   end for
21:    $S' \leftarrow \arg \max_{U \in C} C(U)$ 
22:   return  $f[S']$ ,  $C(S')$ 
23: end function

```

order to introduce some zeroes, we subtract from f^0 its minimum (which amounts to the min-marginal of f^0 onto the empty set). In the example, this subtraction yields function $f^0[\emptyset]$, shown at the top of Figure 5c. Subsequently, it calculates the next remainder f^1 using Equation 1.

After calculating the new remainder f^1 , the new function to approximate, the algorithm proceeds to update the *Zeroes* table along with the C counter. Back to our example, notice that f^1 contains a single tuple with zero cost $t_S = (xyz \leftarrow aab)$. Then, the algorithm calculates the projection of t_S to each row U and sets cell $[U, t_S[U]]$ to *true* in the *Zeroes* table. In the example, the cell for row xy and column aa is set to true in the *Zeroes* table. Moreover, the counter for row xy decreases to record that there is one less non-zero cost tuple. Figure 5a ($i=1$) shows the state of both the *Zeroes* table and the counter vector after iteration $i=1$. In general, for each *new* zero cost tuple t_S , the algorithm checks the *Zeroes* table cell at row U and column $t_S[U]$. If the cell is *false*, it is set to *true* to indicate that the cost of the min-marginals for the tuple will be zero from iteration i onwards. Moreover, the value of the counter of non-zero cost tuples for the min-marginal, $C(U)$, decreases by one.

Once the *Zeroes* table and counters are updated, there are two cases: (1) If all counters' values are zero, it means that the cost for all tuples of all subsequent min-marginals will be zero. Therefore, since it is not possible to extract more information from subsequent min-marginals, the algorithm terminates and returns the list of selected min-marginals so far, $\{f^0[S'_1], f^1[S'_2], \dots, f^m[S'_{m+1}]\}$, as the resulting decomposition. (2) Otherwise, the min-marginal with more non-zero tuples is selected as the best min-marginal, and the algorithm continues.

In the example in Figure 5, all candidate min-marginals (see the rows in table *Zeroes* at iteration $i=1$) contain 3 non-zero tuples. Thus, at the next iteration ($i=2$), the algorithm can randomly choose the marginalization of f^1 onto any pair of variables. Say that the algorithm chooses $\{yz\}$. There-

fore, the selected best min-marginal is $f^1[yz]$, and hence the new remainder f^2 can be computed. After updating the *Zeros* table, there are still two counters larger than zero, as shown in Figure 5a ($i=2$). In our case, the algorithm selects $f^2[xz]$ (discarding $f^2[xy]$), calculates the new remainder f^3 , and updates the *Zeros* table to yield the table in Figure 5a ($i=3$). At this point, since all counters are zero, the algorithm terminates to return the following set of selected best min-marginals as the resulting decomposition:

$$F' = \{f^0[\emptyset], f^1[yz], f^2[xz]\}.$$

On the one hand, notice that the whole procedure—shown in Algorithm 3—never calculates a min-marginal unless it is going to be returned as part of the resulting decomposition. Further, since the maximum number of functions in a decomposition is $\binom{|V|}{r}$, the worst case complexity of calculating the decomposition is $O(\binom{|V|}{r} \exp(|V|))$. On the other hand, the algorithm has to maintain the zeros table, which also has a cost. Note that function `selectBestMarginal` only processes the tuples that are zero in the current iteration and were not zero in the previous iteration⁴. This means that to maintain the table, each tuple will be processed at most once. Since for each tuple we mark each possible min-marginal, the time complexity of maintaining the table is $O(\binom{|V|}{r} \exp(|V|))$, not increasing the overall time complexity.

7. EMPIRICAL EVALUATION

In this section we evaluate the performance of the different function approximation approaches on DIMCTef. For each experiment, we present both the communication savings and increase in overall computational cost with respect to GDL. We choose to track these measures because they are the key ones in constrained environments. For instance, consider a wireless sensor networks setting. Since running out of battery disables a node, battery consumption is probably the most important figure to consider. Therefore, both communication and computation costs are important because they directly determine battery consumption. We estimate the overall computational cost by adding the processing times incurred by each node, while ignoring communication times. Similarly, the overall communication cost can be easily determined by adding the number of bytes of all sent messages.

Because GDL’s communication and computation is mainly determined by the maximum clique size of the computed Junction Tree, experiments are segmented by this parameter. Consequently, both GDL and all DIMCTef approaches use the very same Distributed Junction Tree Generator [15] algorithm to compute JTs. Additionally, notice that the parallelism degree is roughly the same for all algorithms, because it mainly depends on the computed JT. As a consequence, since DIMCTef always communicates less information than GDL, the relative increase in real solving time between GDL and DIMCTef would be lower than the relative increase in overall computation shown in this paper.

Since DIMCTef removes tuples, it generates sparse functions. Sending sparse functions can lead to communication savings, but only if the implementation uses a special codification to transmit these functions. However, exploring the codification of sparse functions was not one of the objectives of this paper. Hence, we simply set a special value as

⁴New zeros can be detected at no cost while computing the combination in line 7.

cost for the filtered tuples, and compressed the messages. Specifically, we chose an Arithmetic Encoder [4] with a Partial Prediction Matching model of 8 bytes. This compression method is known to achieve good compression ratios, so long as its input contains repeated values. The downside is that compressing has a high computational cost, which is considered as part of our overall cost. Regarding GDL, compression hurts because the overall computation increases by an order of magnitude, while communication savings are practically negligible. Therefore, we report GDL results without compressing (following the idea of presenting results for each algorithm in its best possible condition). Likewise, although we tried both the LRE and LMRE metrics for both content-based partitioning and brute-force decomposition, we only report the best results obtained.

We conducted tests with the sensor networks instances from [10], but they were very easy for GDL (5 maximum clique variables). Thus, all approaches lead to the same results, requiring 3 times less communication while maintaining the same computation cost as GDL.

Next, we designed an experiment to measure the methods’ trend as the variables’ arity increases. Thus, it is composed of 35 problems of 20 variables for each domain size, with a random structure of densities ranging from $p=0.1$ to $p=0.3$, where p is the probability of appearance for all edges. Function costs are taken from a normal distribution $\mathcal{N}(0, 1)$, and then made positive by adding its minimum value to each relation. Results in Figure 6a show that top-down approximation methods perform significantly better than bottom-up approximations in the communication front, with nearly constant savings between two and three times better. As expected, the brute force approach is way more expensive computationally than other methods (up to 100 times slower than GDL in the worst case). Nevertheless, zero-tracking’s overall computation cost is just slightly higher (13 times that of GDL at most) than that of the content-based approach (10.5 times), whereas their savings are much larger (110.5 times less sent bytes for zero-tracking against 38.3 times for content-based). Moreover, its savings in communication increase almost 10 times faster than the computational cost.

Then, we conducted a second experiment that measures the trends when the problems’ maximum clique variables increases. Consequently, it contains problems of 20 variables of arity 5 for each maximum clique variables, also with random structures between $p=0.1$ and $p=0.3$, and normal costs. Figure 6b shows that the communication savings increase exponentially for all methods, yet zero-tracking grows at a much faster rate than the others while keeping the computational cost under control.

Finally, the third experiment measures the impact of structure in the problems’ constraint graph. Thus, it contains lattice-structured problems of 25 and 36 variables, leading to JTs of 8 and 10 maximum clique variables. Once again, top-down approximation methods achieve the largest communication savings. In particular, zero-tracking decomposition requires up to 612 times less bytes than GDL in 25% of the clique size 8 problems, while being only 44 times slower.

In summary, top-down approximations result in large communication savings. Additionally, zero-based decomposition remains competitive in computational effort with respect to state-of-the-art approximation methods. Hence, we see zero-based decomposition as the method of choice to optimally solve DCOPs in communication-constrained scenarios.

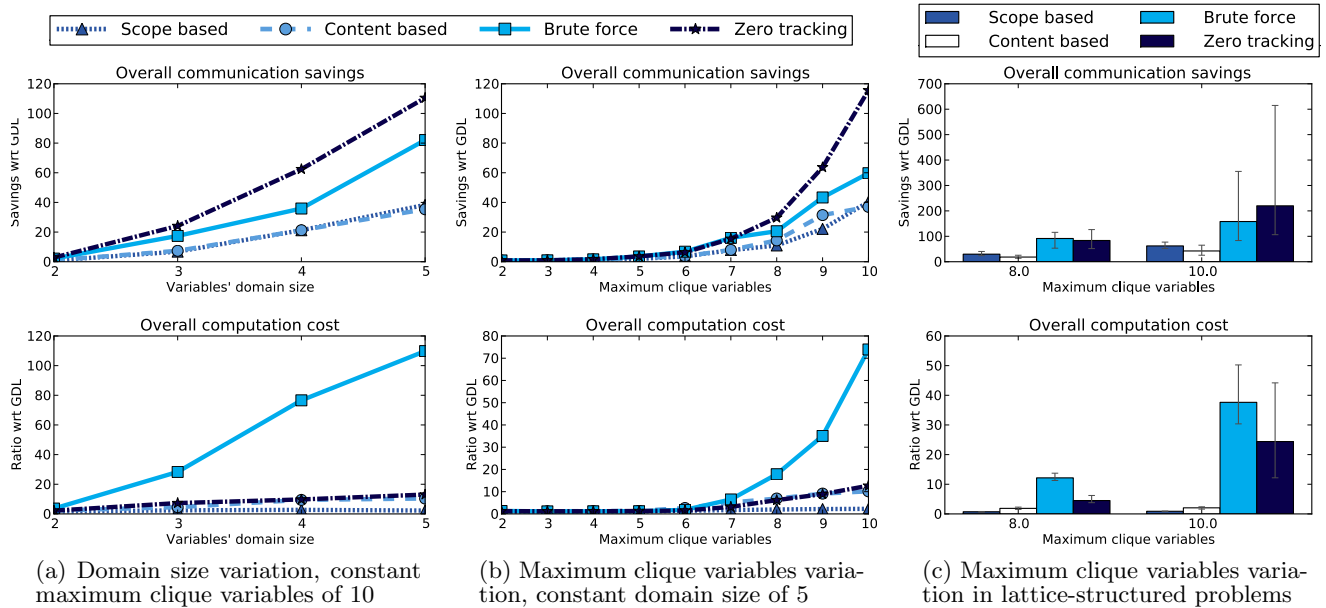


Figure 6: Performance evaluation results.

8. CONCLUSIONS

We addressed the issue of optimal DCOP solving in communication-constrained scenarios, using GDL with function filtering. We first reviewed current state-of-the-art approaches to message approximation, presenting them in a common framework that we named bottom-up approximations. Next, we proposed top-down approximations, a new class of methods designed to reduce communication costs. We then presented two realizations of this novel approach: (1) brute-force decomposition, a naive implementation with high computational cost; and (2) zero-tracking decomposition, which greatly reduces the amount of computation. Finally, we empirically evaluated their performance, showing that top-down approximations always achieve larger communication savings than bottom-up ones. In fact, zero-tracking decomposition does so while keeping the computational cost at bay, becoming the method of choice for optimally solving DCOPs in communication-constrained scenarios.

9. ACKNOWLEDGMENTS

This work has been funded by projects EVE (TIN2009-14702-C02-01 and 02), Agreement Technologies (CONSO-LIDER CSD2007-0022), RECEDIT (TIN2009-13591-C02-02) and Generalitat de Catalunya (2009-SGR-1434 and 2009-SGR-362). Marc Pujol-Gonzalez is supported by the Ministry of Science and Innovation (BES-2010-030466).

10. REFERENCES

- [1] S. M. Aji and R. J. McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.
- [2] I. Brito and P. Meseguer. Distributed cluster tree elimination. In *IJCAI DCR Workshop*, pages 16–30, 2009.
- [3] I. Brito and P. Meseguer. Improving dpop with function filtering. In *AAMAS*, pages 141–148, 2010.
- [4] J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *Communications, IEEE Transactions on*, 32(4):396 – 402, apr. 1984.
- [5] R. Dechter and I. Rish. A scheme for approximating probabilistic inference. *Proceedings of Uncertainty in Artificial Intelligence (UAI'97)*, pages 132–141, 1997.
- [6] B. Faltings, D. Parkes, A. Petcu, and J. Sheidman. Optimizing streaming applications with selfinterested users using M-DPOP. In *COMSOC'06*, pages 206–219, 2006.
- [7] A. Farinelli, A. Rogers, and N. Jennings. Bounded approximate decentralised coordination using the max-sum algorithm. In *IJCAI DCR Workshop*, pages 46–59, 2009.
- [8] F. V. Jensen and F. Jensen. Optimal junction trees. In *UAI*, pages 360–366, 1994.
- [9] C. Kiekintveld, Z. Yin, A. Kumar, and M. Tambe. Asynchronous algorithms for approximate distributed constraint optimization with quality bounds. In *AAMAS*, pages 133–140, 2010.
- [10] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking dcop to the real world: Efficient complete solutions for distributed multi-event scheduling. In *AAMAS*, pages 310–317, 2004.
- [11] P. J. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. Adopt: asynchronous distributed constraint optimization with quality guarantees. *Artif. Intell.*, 161(1-2):149–180, 2005.
- [12] M. A. Paskin, C. Guestrin, and J. McFadden. A robust architecture for distributed inference in sensor networks. In *IPSN*, pages 55–62, 2005.
- [13] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, pages 266–271, 2005.
- [14] E. Rollon and R. Dechter. New mini-bucket partitioning heuristics for bounding the probability of evidence. In *AAAI*, pages 1199–1204, 2010.
- [15] M. Vinyals, J. A. Rodríguez-Aguilar, and J. Cerquides. Constructing a unifying theory of dynamic programming dcop algorithms via the generalized distributive law. *JAAMAS*, pages 1–26, 2010.
- [16] M. Vinyals, J. A. Rodríguez-Aguilar, and J. Cerquides. Egalitarian utilities divide-and-coordinate: Stop arguing about decisions, let's share rewards! In *ECAL*, pages 1025–1026, 2010.
- [17] W. Zhang, G. Wang, Z. Xing, and L. Wittenburg. Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks. *Artificial Intelligence*, 161(1-2):55–87, 2005.

Agent-Based System Development II

AgentScope: Multi-Agent Systems Development in Focus

Elth Ogston
Delft University of Technology
e.f.y.ogston@tudelft.nl

Frances Brazier
Delft University of Technology
f.m.t.brazier@tudelft.nl

ABSTRACT

Multi-agent systems form the basis of many innovative large-scale distributed applications. The development of such applications requires a careful balance of a wide range of concerns: a detailed understanding of the behaviour of the abstract algorithms being employed, a knowledge of the effects and costs of operating in a distributed environment, and an expertise in the performance requirements of the application itself. Experimental work plays a key role in the process of designing such systems. This paper examines the multi-agent systems development cycle from a distributed systems perspective. A survey of recent experimental studies finds that a large proportion of work on the design of multi-agent systems is focused on the analytical and simulation phases of development. This paper advocates an alternative more comprehensive development cycle, which extends from theoretical studies to simulations, emulations, demonstrators and finally staged deployment. AgentScope, a tool that supports the experimental stages of multi-agents systems development and facilitates long-term dispersed research efforts, is introduced. AgentScope consists of a small set of interfaces on which experimental work can be built independently of a particular type of platform. The aim is to make not only agent code but also experimental scenarios, and metrics reusable, both between projects and over simulation, emulation and demonstration platforms. An example gossip-based sampling experiment demonstrates reusability, showing the ease with which an experiment can be defined, modified into a comparison study, and ported between a simulator and an actual agent-operating system.

Categories and Subject Descriptors

C.2.4 [Computer-Communication Networks]: Distributed Systems—*Distributed Applications*

General Terms

Experimentation

Keywords

Multi-Agent Systems Development

1. INTRODUCTION

Agents, unlike passive nodes in traditional computer systems, analyse and react to their surroundings, autonomously making decisions and adapting to their environment. These properties pose a unique challenge in the design of distributed systems. Such knowledge intensive activities are predicated on the availability of information. **Cite as:** AgentScope: Multi-Agent Systems Development in Focus, E. Ogston and F. Brazier, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 389-396.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

As systems scale up and become more decentralised the costs of gathering information becomes an important consideration in an agent's design. Increases in complexity which are beneficial in theory may not prove cost effective in application. Or agent algorithms may set requirements that are beyond the resources of an underlying computing system.

For agent algorithms in which information obtained from the environment in a large-scale distributed system is crucial, design is not limited to the theoretical or conceptual stage of development, but is a continuous process that spans into an experimental implementation, testing and comparison process. Procedures that provide information services need to be carefully matched, and perhaps customised. The performance of an algorithm must be examined in combination with the services it uses. Possible designs must be compared in the intended application setting, and improvements in effectiveness weighed against increased cost.

This process of designing practical multi-agent systems can be viewed as an incremental development cycle, moving from theoretical studies to experimental simulations, emulations, and demonstrators, and finally to staged deployment. Experimental work plays an important role in the process. Simulation, emulation and demonstration experiments allow key aspects of system behaviour to be examined in detail in a controlled environment. Correctness and performance of algorithm implementations can be confirmed, and alternative algorithms can be carefully compared.

In contrast to this comprehensive view of the development cycle, a survey of recent papers finds that experimental work on multi-agent systems is heavily weighted towards the simulation stage. A large proportion of works appear to be strongly influenced by methodologies that view system design as occurring primarily in the theoretical or analytical stage of development.

AgentScope is a tool that supports the experimental stages of multi-agents systems development and facilitates long-term dispersed research efforts. AgentScope defines a set of generic interfaces for (1) networked communication between agents, (2) measurement and analysis of agent behaviour, and (3) the setup of experimental scenarios. The AgentScope interfaces allow protocols and experiments written for one type of platform, for instance a simulation environment, to be easily ported to other types of platforms, such as emulation environments, as research progresses through the development cycle. Real agent environments, such as the AgentScope middleware platform [15], can also be used, simplifying the transition between the experimental and deployment stages of the development cycle. AgentScope further enables the publishing of experiments, making it easier to compare algorithms with previous work, and reuse experimental scenarios and metrics.

The aim of the AgentScope project is to allow researchers to view experimental work from a more ambitious perspective than

the testing of hypotheses for a single publication or project. AgentScope views experiments not as a one-off effort by a single researcher but as long-term work by many researchers. AgentScope gives support for measurement, analysis and scenario development equal priority to support for algorithm development. AgentScope promotes the creation of code whose use is not restricted to studying only the aspects of behaviour currently of interest, but that can be carried through a series of experiments, and be used throughout the development cycle.

This paper discusses the experimental stages of multi-agent systems development (Section 2), extending the argument for a comprehensive agents development cycle put forward in [11]. It further surveys existing experimental work (Section 3), and presents the AgentScope interfaces (Section 4). An example in Sections 5 and 6 demonstrates the ease with which experiments can be developed, extended and ported between different types of platforms using AgentScope. Sections 7 and 8 present conclusions and summarise ongoing and future work.

2. MULTI-AGENT SYSTEMS DEVELOPMENT

AgentScope considers experiments not as one off efforts by a single researcher but as part of long-term work by many researchers. From this point of view an experiment is a small step in a larger application *development cycle*. In multi-agent systems complex communication processes underlie important high-level procedures such as coordination, negotiation, collaboration, and coalition formation, to name a few. The deployment of such procedures in applications is more complex still. Ideas developed in theory rarely translate directly to deployment. Instead an intricate development process must often be followed to bridge the gap between theory and application.

In AgentScope, rather than placing agents and their overall behaviour foremost, the core abstraction is a *protocol*; a distributed set of components that collectively provide a specific distributed service. Agents are made up of a set of protocols that support their core behaviour. This change in focus views a multi-agent system not only from an agent's perspective, but also as a distributed system. Protocols map the abstract services employed by an agent to concrete implementations of distributed algorithms.

The development path between abstract agent design and full scale application deployment is divided into phases, a simulation stage, an emulation stage and a demonstration stage. Experiments in each stage single out the behaviour of specific protocols and test how well protocols combine to provide full agent functionality.

2.1 Protocols

A protocol supports a distributed application by providing a basic abstract service. For instance sampling, aggregation, dissemination, resource allocation, clustering, directories, and search are common distributed tasks that can be viewed as services used by higher-level applications. While each of these tasks is a relatively simple concept, the many factors that must be considered in a distributed environment often result in intricate communication and coordination patterns. Implementations require careful attention to detail and testing. Separating such tasks out simplifies testing, improves code reusability, and allows implementation details to be hidden to a great extent from the application that uses them.

Protocols consist of two parts: the abstract *algorithm* that is used to achieve the desired function, and the *implementation* of that algorithm. Algorithms can be, and often are, studied separately from an implementation. However, in an application context the two parts are best studied in combination. Implementation details can have a major impact on applicability. Assumptions made about the

underlying system can lead to performance under different environmental conditions varying widely from that expected. Theoretical analysis often makes use of abstract concepts that may not have straightforward manifestations: the uniformity of random samples, the existence of clustering criteria, the presence of common parameters agreed among autonomous entities, for instance. Practical implementations often contain non-deterministic or heuristic behaviour that does not lend itself well to theoretical analysis.

2.2 Experimental Stages of Development

Protocol development commonly goes through three experimental phases between theory and end application: 1. simulation, 2. emulation, 3. use in application demonstrators. Each phase focuses on testing and improving a different aspect of a protocol implementation. Simulations test the *functionality* of a protocol to show that the basic algorithmic design is correct and complete. Emulations test functionality and performance in a *distributed setting*. Demonstrators test if the performance characteristics of a protocol, and assumptions about the system it relies on, *match the application* or class of applications in which it will eventually be used. The main practical difference between the phases is the degree to which aspects of the eventual application environment are replaced by abstract or simplified models.

Simulations involve building up a detailed understanding of an algorithm's basic functionality. The aim is often to confirm or enhance a theoretical analysis. Abstract models of the eventual application environment are used to focus in on key theoretical behaviours. Full protocol details are often not of interest. Since algorithms are usually previously untested, detailed debugging can be involved. Simulations are therefore best suited to environments that run on a single machine, using a simplified model of the eventual distributed setting. This allows for recording and analysing large amounts of data, such as the internal state of all agents, and for stopping the experiment clock to take a single synchronised snapshot of the system to test if global invariants are upheld.

Emulations are characterised by the use of a real distributed environment to confirm protocol functionality when actual communication characteristics are taken into account. Emulations are concerned with the implications of parallelisation. Simulations often take into account the location and replication of data and what messages need to be passed between agents. Emulations further consider timing and synchronisation, bandwidth requirements, the effect of communication latency and errors, and so forth. While these concerns can be partly tested in simulations, the use of an actual network is often simpler than the effort required to model it. The use of a network analogous to that used by the intended end application avoids the need to identify and model every aspect that may be of importance.

Demonstrators take a step towards testing an algorithm in a real application setting. Simulations and emulations usually focus on the full range of algorithm behaviour tested on abstract data sets. Demonstrators add models of intended uses for large-scale distributed systems based on expert analysis and real data. They test if an algorithm is suited for a specific purpose as opposed to testing its generic behaviour.

3. RELATED WORK

Experimental work is an integral part of many Agents research projects. In order to obtain a rough picture of the experimental tools and methods used in recent work a survey was made of the 61 papers presenting original work published in the Journal of Autonomous Agents and Multi-Agent Systems (JAAMAS) in 2009 and 2010 [1]. JAAMAS covers a wide range of topics, 37 of the pa-

pers surveyed presented experimental results, of these 18 ran experiments on distributed, potentially large-scale multi-agent systems. A further examination is made of these 18 papers as they represent the type of experimental work directly targeted by AgentScope.

The vast majority of work falls in the simulation phase of the development cycle. None of the 18 papers examined deployed applications, only two were based on deployed demonstrators. Of the remaining 16 all ran simulations. None of these simulations could be characterised as emulations, running on a real network. Though six papers ran more advanced simulations that modelled messages, only two of these measured communications costs and only one of those two stated that it was run in real time rather than being rounds based. Only three of the simulations used a scenario based on measurements of actual systems.

It is surprising that so little experimental work falls within the later stages of the development cycle, especially given that JAA-MAS papers generally represent the more mature projects within the field. Given the importance of considering application characteristics it is disappointing that there is little work representing deployed applications, or scenarios based on expert analysis and measurements of existing systems. This imbalance may reflect the influence of development methodologies that focus on the theoretical side of Agents research, considering the bulk of important design decisions to occur in the early analytical stages of development. Methodologies, such as Gaia [16], which place an abstraction barrier between design and implementation, make an implicit assumption that implementation and application details will have little effect on system design.

Reusability is a key factor in supporting the wider spread adoption of a more comprehensive development methodology. The reuse of agent components, as discussed in [2], lowers the cost of extending previous work and running comparisons studies. We found little evidence of the reuse of agent components in the 18 works surveyed. While all the papers discussed related algorithms, only half did experimental comparisons. Four of these mentioned either needing to re-implement previous work or not being able to do a further comparison because of a lack of code.

Standard experimental platforms further improve reusability. Examining platform reuse found that of the 18 selected papers, four used existing published simulators, and two used previously published demonstrators. For five papers a custom simulator was developed, seven papers did not state what simulator was used. A good variety of simulators are in fact available [4, 6, 10, 13]. The JamesII project [5] shows a common interface can be used to allow simulations to be ported between simulators. AgentScope follows this approach, extending reusability by emphasising the desirability of porting experiments between different types of platforms [14] rather than focusing on simulations.

A third aspect of reusability concerns the development of common scenarios, metrics and experiments. In the papers surveyed, scenario and metric reuse was more common than direct reuse of code: nine papers were based on previously developed scenarios, and three extended previous scenarios. Twelve papers used previously published metrics, and one used an extension of a previous metric. Six papers developed custom scenarios, five of these created custom metrics. JamesII [5] includes mechanisms to reuse experiments. In AgentScope the interfaces for implementing metrics and building experimental scenarios are given equal importance to the interface on which agents are built. The intention is to encourage the creation of standard testbeds, which make input from domain experts, and data sets more accessible. The success of agent competitions gives an indication of the value of this approach. Three of the four simulators that were explicitly reused

in the papers surveyed were from agents competitions: Robocup [9], Robocup Rescue [7], and the Trading Agents Competition [8]. It is perhaps significant that these simulators are associated with full test beds, including scenarios and metrics, and for which code for previous solutions is available. AgentScope aims to improve the extendability of this approach by enabling the creation of more generalised scenario libraries.

4. THE AGENTSCOPE INTERFACES

AgentScope gives support for measurement, analysis and scenario development equal priority to support for algorithm development. It defines three interfaces that separate out the core behaviours needed to implement distributed protocols and run experiments: (1) Networked Communication (2) Measurement and Analysis and (3) Experimental Control. The Protocol Network Interface provides the methods and classes needed for agent protocol instances to communicate with each other. Protocol code written on the Protocol Network Interface can be designed to be used throughout the whole development cycle. The Measurement and Analysis Interface defines a generic method of recording protocol behaviour during an experiment. The specific measurements of interest generally change between experiments. The Measurement and Analysis Interface is designed to allow measurement code for a protocol to be easily swapped out and replaced depending on the experiment being run. The Experimental Control interface defines a generic method of setting up experiments so that an experimental scenario can be easily reused, or ported between platforms.

AgentScope promotes the creation of code whose use is not restricted to studying only the aspects of behaviour currently of interest, but that can be carried through a series of experiments, and be used throughout the development cycle. AgentScope interfaces are intentionally minimal. The restrictive interfaces encourage developers to use methods that are as generic as possible to the many environments encountered during the development cycle.

AgentScope interfaces are two sided. On one side they provide a generic set of classes and methods for use by protocol developers. On the other side they provide a set of classes and methods that a *backend* must implement in order to run AgentScope based protocols and experiments. In a backend, a set of “adaptor” classes map a platform’s functionality to that required by AgentScope. These adaptor classes can provide additional methods that allow experiments to access functionality that is specific to a particular platform. In the following sections we give details of the protocol developer side of each interface. In Section 6 we discuss backend adaptors. The AgentScope interfaces are implemented in Java.

4.1 Protocol Network Interface

The Protocol Network Interface is a stand alone package defining the basic classes and interfaces that are needed to write protocols. The core abstraction is a “protocol”: an implementation of a distributed algorithm that provides a particular service to “agents” located on nodes in a distributed system. Each participating agent runs an instance of the protocol, and collectively these protocol instances provide the service to their agents. For instance a directory protocol might maintain a list of active agents, an aggregation protocol might calculate the sum of a given dynamic variable or a load balancing protocol might distribute a set of tasks between agents.

Protocol instances each have an individual “address”, which allows them to communicate with each other by exchanging “messages”. They are triggered into action on receipt of a message, in response to “events” that are set to occur at a particular time (according to a local “clock”), or in response to a request made by their agent. Protocols also have “names” to allow particular in-

Table 1: The Protocol Network Interface

METHODS OF PROTOCOL CLASS
<i>Protocol</i> (Agent myAgent, String name) <i>sendMessage</i> (Message m) <i>receiveMessage</i> (Message m) <i>scheduleEventAt</i> (String note, long time) <i>scheduleEventWithDelay</i> (String name, long timeFromNow) <i>triggerEvent</i> (Event p) <i>getAddress</i> () returns Address <i>getName</i> () returns String <i>getClock</i> () returns Clock <i>start</i> ()
METHODS OF MESSAGE CLASS
<i>Message</i> (Address to, Address from) <i>to</i> () returns Address <i>from</i> () returns Address
METHODS OF EVENT CLASS
<i>Event</i> (Address addr, String name, long time) <i>getName</i> () returns String <i>getAddress</i> () returns Address <i>getTime</i> () returns long
METHODS OF AGENT INTERFACE
<i>addProtocol</i> (Protocol p, String name) returns Address <i>getProtocol</i> (String name) returns Protocol <i>sendMessage</i> (Message m) <i>scheduleEvent</i> (Event p) <i>getClock</i> () returns Clock
METHODS OF ADDRESS INTERFACE
<i>sameAs</i> (Address a) returns boolean <i>sameAgent</i> (Address a) returns boolean
METHODS OF CLOCK INTERFACE
<i>currentTime</i> () returns long <i>unitsPerSecond</i> () returns double

stances of a given protocol to be located when the address of an agent is known, but not the specific address of the named protocol on that agent.

The Protocol Network Interface consists of three classes - Protocol, Message and Event - and three interfaces - Agent, Address and Clock (Table 1). The classes represent the main objects that a protocol manipulates, the interfaces represent supporting concepts from the system the protocol exists within. A protocol implementation is written by subclassing the Protocol class. Subclasses of Message and Event are used to represent protocol specific messages and events. The Agent interface provides methods for sending and receiving messages and setting and receiving events.

4.2 Measurement and Analysis Interface

The Measurement and Analysis Interface provides classes in which to define: (1) the data that should be recorded about a protocol's behaviour during an experimental run, and (2) how that data should be manipulated to provide the final output of the experiment. The core abstraction is a "logbook" - essentially a blank space in which a protocol instance records information, along with methods that define how to process that information. Logbooks are located on Agents. Each protocol instance can have one or more logbook instances associated with it.

The Measurement and Analysis Interface consists of two classes, LogBook and its subclass ActiveLogBook, and one interface, Logger (Table 2). A LogBook stores and manipulates the raw data recorded during an experiment. A Logger interacts with the backend to aggregate the recorded data into a final single location. In order to write an experiment a designer must subclass LogBook to specify the data of interest, and methods for analysing it.

There are two modes in which a logbook can operate - passive and active. When a logbook is passive, the protocol code specifies the data to record. When a logbook is active the logbook code

Table 2: The Measurement and Analysis Interface

METHODS OF LOGBOOK CLASS
<i>LogBook</i> (String name) <i>aggregate</i> (LogBook moreData) <i>clear</i> () returns LogBook <i>writeData</i> (File directory) <i>combineData</i> (File[] inputDirs, File outputDir) <i>getName</i> () returns String
METHODS OF ACTIVELOGBOOK CLASS
<i>ActiveLogBook</i> (Protocol p, String name) <i>recordProtocolState</i> ()
METHODS OF LOGGER INTERFACE
<i>addLogBook</i> (LogBook l)

specifies the data to record, a protocol need not contain specific logging code or know of the log's existence. Active logging is better suited when fully experiment-generic protocol code is desired. Passive logging is better suited to detailed measurements that involve recording events as they occur.

The more basic class, LogBook operates in passive mode. In passive mode, a protocol instance must be informed of the a logbook's existence, for example by the logbook instance registering as a listener to the protocol instance. The protocol code specifies what information to store. ActiveLogBook subclasses LogBook to add functionality for recording data independently of a protocol implementation. In active mode a logbook instance holds a pointer to a protocol instance. The data to recorded is specified within the logbook code. When triggered, for instance by a periodic signal from the logger, the logbook calls methods on the protocol to extract data.

Logbooks store data in a distributed manner within the agents which they are monitoring. A backend adaptor implementation of the Logger interface specifies details of how individual logbooks are routed by the experimental system to produce the final output of an experiment at a central location. The Logger combines data from the individual agent logs into a single central logbook instance. The LogBook.aggregate() method is used to specify how data from individual logs should be combined. The LogBook.writeData() method specifies how the final results for an experiment should be recorded to a specified a directory, for instance in the form of graphs of tables. The LogBook.combineData() method further specifies how the output produced by the writeData() method for several different experimental runs can be combined into a single set of results.

The AgentScope toolkit contains a supporting package of classes for storing, manipulating, and analysing data and drawing graphs and tables with which logbooks can be implemented. It also contains a set of generic logbooks, for instance for recording, analysing and graphing series or time sequences of values. Additionally, a logger implementation that is built entirely on the Protocol Network Interface is provided. In this case the logger is a protocol that uses messaging to move data, and events to trigger active log functions. For backends that contain specific logging support, such as synchronised triggers for taking system snapshots, a logger implementation can be built that makes optimal use of that support.

4.3 Experimental Control Interface

The Experimental Control Interface provides a means of organising the setup of an experiment in a flexible manner. Both the need to change experiment configuration and the need to port experiments between backends are taken into consideration. The setup and running of an experiment is divided into several parts. First, "experiments" are distinguished from "trials". An experiment is a full experimental scenario while trials are single runs of that sce-

Table 3: The Experimental Control Interface

METHODS OF ENVIRONMENT INTERFACE
<code>runTrial(int numAgts, Initializer i, LogBook l) returns LogBook</code>
METHODS OF EXPERIMENT INTERFACE
<code>run(File outputDir)</code>
<code>runTrial(Trial t, File outputDir)</code>
<code>combineOutput(File[] inputDirs, File outputDir)</code>
METHODS OF INITIALIZER CLASS
<code>Initializer(Trial trial)</code>
<code>initializeAgent(Agent a, Logger l)</code>
<code>finalizeSetup() returns boolean</code>
<code>getTrial() returns Trial</code>
METHODS OF TRIAL CLASS
<code>Trial(String name, int numAgents, int trialLength)</code>
<code>getNumAgents() returns int</code>
<code>getName() returns String</code>
<code>getTrialLength() returns int</code>

nario. Second, the “core” experimental scenario is separated from the parts of the experiment that may be configured differently in different trials.

The division of purpose is represented by the Experimental Control Interface classes and interfaces: (1) an Experiment provides the interface through which to run a series of trials, (2) an Environment captures information about the setup of the backend environment in which the scenario is to be run, (3) an Initializer stores information on the setup of agents for a core experimental scenario, and (4) a Trial stores information on the configuration of a particular trial. (Table 3)

An Experiment provides an interface for running a trial or series of trials. Within an Experiment a designer specifies the Environment or Environments to use, Initializers, a core series of trials to run, and how the output of a series of trials should be combined to give the final experimental results. Experiment provides the basic methods on which user interfaces can be built. The AgentScope toolkit provides a simple command line UI, a web-based UI and a generic file manager that provides methods for running complex series of trials and storing them to a standard directory structure.

An Environment is the part of the adaptor for a given backend with which an Experiment interacts. An Experiment runs a trial on a backend by calling that backend’s Environment.runTrial() method. The runTrial() method takes an Initializer as input which defines the generic agent setup. Running an experiment on a new backend only requires a switching Environments.

An implementation of Initializer specifies how agents should be set up to run a given scenario, in a backend independent manner. At the start of an experiment a backend creates empty agents, which are then passed to the initializer. The initializer adds the required protocols and logbooks to the agents and sets up the initial connections between them.

A Trial keeps track of trial specific configuration details. An Initializer specifies the setup of a generic scenario, it can be given a Trial instance to query for parameters that may vary between trials. Trials can for example be used to vary the values of variables, the number of agents, protocol implementations used, initial connections between agents, experiment event series, logbooks used, input datasets, etc. An experiment designer can thus define the fixed and variable parts of an experimental setup through defining a core Initializer and a Trial or set of Trials specifying what may change between experiment trials.

5. SAMPLING EXPERIMENT EXAMPLE

Writing an experiment in AgentScope requires developing a protocol to be tested, logbooks to measure its performance, and an experimental setup in which to run. This process is demonstrated for

the implementation and testing of a basic sampling protocol, SimpleGossip. A *sampling service* is a degenerate form of directory service that when queried returns a randomly chosen address of an agent in the system. SimpleGossip is an unsophisticated gossip-based sampling protocol. In gossip-based sampling each agent maintains a cache of items. Each item stores the address of an agent. Pairs of agents periodically “gossip” with each other, exchanging items from their caches. Repeated gossiping creates a continuous mixing procedure in which items become spread randomly throughout the agent caches. When an agent needs a random address, the sampling protocol returns a random item from its cache. Ideally the samples returned by a sampling protocol will follow a uniform random distribution. The exact gossiping procedure, as well as a protocol’s response to node churn, message loss and other underlying system characteristics determine the degree to which it meets this requirement. A detailed discussion of gossip-based sampling is presented in [12].

The following sections provide the bulk of the code needed to implement and test SimpleGossip using AgentScope and some convenience classes from the AgentScope toolkit. First a basic simulation experiment is developed in Sections 5.1-5.3. In Section 5.4 this experiment is extended to a more complex setting involving node churn, and a comparison to an existing protocol is performed. In Section 6 a further version of the experiment is run on a full distributed agent middleware platform, AgentScope [15].

5.1 Protocol Development

The core SimpleGossip protocol, given in the code lines 1-45, is developed on the Protocol Network Interface. SimpleGossip subclasses protocol. Each SimpleGossip instance contains an address-item cache of size C (line 4). Periodically, with some interval t seconds (lines 34-37), each instance chooses a gossip partner and sends it g of the items from its cache (lines 13-16). This agent responds by returning g items from its own cache (lines 17-23). Notice that some time can pass between when a request is sent and the reply is received. No attempt is made to manage the ordering of gossips. Subsequent incoming gossips can be handled in this interval. Nor does SimpleGossip keep track of the requests it makes, or notice when a gossip fails and no reply is received. An agent joins a SimpleGossip protocol by filling its cache with items for its own address, then initiating a gossip with a bootstrap agent already in the protocol (lines 25-31).

```

1 public class SimpleGossip extends Protocol{
2   int t=1; int C=25; int g=3;
3   Address bootstrapAgent;
4   ItemCache theCache;
5   public SimpleGossip(Agent myNode, Address bootstrapAgent){
6     super(myNode, "SimpleGossip");
7     this.bootstrapAgent = bootstrapAgent;
8     theCache = new ItemCache(C);
9   }
10  public Address getRandomAddress(){
11    return theCache.getRandomItem().getAddress();
12  }
13  protected void initiateGossip(Address partner){
14    Item[] toSend = theCache.removeItems(g);
15    sendMessage(new GossipRequestMessage(partner, getAddress(), toSend));
16  }
17  protected void receiveGossipMessage(GossipMessage m){
18    theCache.addAll(m.getItems());
19    if(m.isRequest()){
20      Item[] toSend = theCache.removeItems(g);
21      sendMessage(new GossipReplyMessage(m.from(), getAddress(), toSend));
22    }
23  }
24  @Override
25  public void start(){
26    theCache.createAndAddOwnItems(C);
27    scheduleEventWithDelay("GOSSIP", t);
28    if (bootstrapAgent != null) {
29      initiateGossip(bootstrapAgent);
30    }
31  }
32  @Override
33  public synchronized void triggerEvent(Event p){
34    if (p.getName().compareTo("GOSSIP") == 0) {
35      scheduleEventWithDelay("GOSSIP", t);

```

```

36     initiateGossip(getRandomAddress());
37 }
38 }
39 @Override
40 public synchronized void receiveMessage(Message m) {
41     if (m instanceof GossipMessage) {
42         receiveGossipMessage((GossipMessage) m);
43     }
44 }
45 }

```

5.2 Basic Experimental Setup

In the AgentScope Control Interface an initialiser is used to define the basic setup of agents in an experimental scenario (lines 46-61). For the sampling scenario this involves creating a SimpleGossip protocol instance for each agent, and informing each protocol of the bootstrap agent's address (line 51).

```

46 public class SamplingInitializer implements Initializer {
47     private int agentCount = 0;
48     Address bootstrapAgent = null;
49     @Override
50     public void initializeAgent(Agent a, Logger l){
51         SimpleGossip p = new SimpleGossip(a, bootstrapAgent);
52         if(bootstrapAgent == null){
53             bootstrapAgent = p.getAddress();
54         }
55         agentCount++;
56     }
57     @Override
58     public boolean finalizeSetup(){
59         return agentCount == getTrial().getNumAgents();
60     }
61 }

```

The sampling experiments are run on the Platform 9 3/4 simulator (line 65), the custom simulator used in [12]. A subclass of the Experiment class is used to define backend setup (lines 62-72). In the initial experiments specific SamplingProtocol parameters are not varied, so the the basic Trial class can be used directly.

```

62 public class SamplingExperiment extends Experiment {
63     @Override
64     public void runTrial(Trial t, File outputDir){
65         Environment env = new P934Environment();
66         env.setTrialLength(t.getLength());
67         Initializer init = new SamplingInitializer();
68         LogBook log = new DoubleValueLog();
69         LogBook results = env.runTrial(t.getNumAgents(), init, log);
70         results.writeData(outputDir);
71     }
72 }

```

5.3 Measurement

Measurement involves creating a set of logs to record metrics of interest to an experiment. The uniformity of the samples returned by SimpleGossip can be tested by using sampling to estimate the total number of agents, N , in the system [12]. Using the inverted birthday-paradox, let x be the total number of items seen before two items for the same agent are detected. The estimate of N is then $x^2/2$. Agents can make repeated estimates of the network size by watching the stream of incoming items produced by SimpleGossip. The statistical accuracy of this estimate gives a measure of the uniformity of the distribution from which the samples are drawn.

A listener model is used to allow objects to register to be notified each time SimpleGossip receives a new item. A class, SizeEstimate, watches the item stream and generates size estimates using the inverted birthday-paradox method. In turn SizeEstimate informs listeners registered with it each time a new estimate is generated. These estimates are recorded using the DoubleValueLog class, from the AgentScope toolkit (lines 73-113). DoubleValueLog uses the DataSet class from the AgentScope toolkit to store and analysis values (line 74). The Chart class from the AgentScope toolkit is used to produce graphs of these values (lines 92-104).

```

73 public class DoubleValueLog extends LogBook implements ValueEventListener<Double>{
74     DataSet values;
75     public DoubleValueLog(String name){
76         super(name);
77         values = new DataSet();
78     }

```

```

79     @Override
80     public synchronized void newValue(Double v) {
81         values.addValue(v);
82     }
83     @Override
84     public synchronized void aggregate(LogBook moreData) {
85         if (moreData instanceof DoubleValueLog) {
86             DoubleValueLog l = (DoubleValueLog) moreData;
87             values.addValue(l.getValues());
88         }
89     }
90     @Override
91     public void writeData(File directory){
92         Chart c = values.graphDistribution(1, getName());
93         c.saveImageAndSerializedChart(directory, getName());
94     }
95     @Override
96     public void combineData(File[] dirs, File outputDirectory) {
97         LineChart outputChart =
98             new LineChart(getName(), "Size Estimate", "Number of Occurrences");
99         for(File f: dirs){
100             File nextFile = new File(f, getName() + ".ser");
101             LineChart c = (LineChart) Chart.readSerializedChart(nextFile);
102             outputChart.addDataSeries(f.getName(), c.getDataSeries("dist"));
103         }
104         outputChart.saveImageAndSerializedChart(outputDirectory, getName());
105     }
106     @Override
107     public synchronized LogBook clear(){
108         DoubleValueLog l = new DoubleValueLog(getName());
109         l.values = values;
110         values = new DataSet();
111         return l;
112     }
113 }

```

In order to specify the measurements to be made during the experiment, the initializeAgent() method of SamplingInitializer (lines 50-56) is extended (lines 116-124) to create a SizeEstimate (line 118) and log (line 120) on each agent, and to register each log with the platform logger (line 122). The logger takes care of transferring the values recorded in each agent log to a central log. To gather data the logger calls the clear() method (lines 107-112) on each registered agent log. The logger gives this data to the central log through the aggregate() method (lines 84-88). This central log is specified in Experiment.runTrial() to be another instance of DoubleValueLog (line 69).

```

115     @Override
116     public void initializeAgent(Agent a, Logger l){
117         SimpleGossip p = new SimpleGossip(a, bootstrapAgent);
118         SizeEstimate e = new SizeEstimate();
119         p.addListener(e);
120         DoubleValueLog log = new DoubleValueLog("SizeEstimate");
121         e.addListener(log);
122         l.addLogBook(log);
123         ...
124     }

```

Experiment.runTrial() also specifies what should be done with the central log at the end of an experiment (line 70). The DoubleValueLog write method produces a chart of the distribution of values recorded (lines 91-94).

An experiment series is defined in the run method of the SampleExperiment class (lines 126-131). Two trials are specified, one with 100 agents and one with 1000 agents. Both are run, and the output is combined using the DoubleValueLog.combine() method (lines 133-135). The FileManager class from the AgentScope toolkit is used to specify the directory structure for storing results. The FileManager runTrial() and combineData() methods simply call the corresponding Experiment methods with appropriate File arguments.

```

125     @Override
126     public void run(File outputDir) {
127         FileManager fm = new FileManager(this, outputDir);
128         fm.runTrial(new Trial("SimpleGossip100", 100, trialLength));
129         fm.runTrial(new Trial("SimpleGossip1k", 1000, trialLength));
130         fm.combineOutput();
131     }
132     @Override
133     public void combineOutput(File[] inputDirs, File outputDir) {
134         (new DoubleValueLog("SizeEstimate")).combineData(inputDirs, outputDir);
135     }

```

The final output of the experiment is shown in Figure 1. The figure shows the distribution of size estimates made by all agents during a run. Vertical lines mark the average. With 100 agents SimpleGossip performs reasonably well, on average estimating the

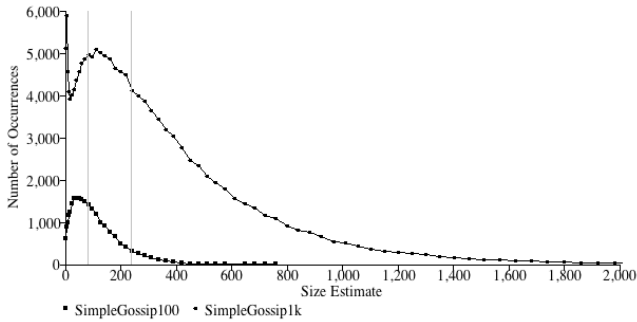


Figure 1: Size estimates: basic simulations.

system size to be 82.6. With 1000 agents however it gives an average size estimate of 238.7 showing that SimpleGossip is not a precise sampling protocol.

5.4 Extending the Experiment

The experiment described above can be easily extended. Alternative sampling protocols can be tested against SimpleGossip to directly compare performance. Additional metrics can be recorded. Or the environment in which the scenario is run can be modified.

In order to compare SimpleGossip to alternative sampling protocols a `SamplingTrial` class is created which specifies which protocol to use through a `getSamplingProtocol()` method. The `SamplingInitializer` calls this method when setting up an agent (line 51). A library version of the Eddy protocol, described in [12], can then be run in the scenario developed for SimpleGossip simply by creating and using a corresponding `SampleTrial`.

Analysing different aspects of SimpleGossip’s performance only requires adding or modifying logbooks. For instance, to record how the size estimates produced by the protocol change over time, the `DoubleValueLog` used by the agents (line 120) can be replaced by a `TimeStampedValueLog` and the `DoubleValueLog` used by the `SamplingExperiment` can be replaced with a `TimeStepsValueLog` (line 69). Both of these are generic logs from the AgentScope toolkit. `TimeStampedValueLog` extends `DoubleValueLog`, recording the time at which each value is logged, and `TimeStampedValueLog` uses these times to divide values into time steps.

Finally, the environmental conditions in which SimpleGossip operates can be modified by configuring the Platform 9 3/4 backend. The simulator can be set to implement churn, causing agents to fail over time and adding new agents by calling the `P934Environment.setChurn()` method after creating the environment in `SamplingExperiment.runTrial()` (line 65).

Figure 2 shows the end result of these changes. The figure compares the ability of SimpleGossip and Eddy to estimate system size over time in a system with churn and an average size of 1000 agents. It highlights an important failing of SimpleGossip, as the agent set changes the item set is not adapted, resulting in the quality of samples degrading over time. Eddy shows that a more complex protocol can manage the item set in a way that allows it to estimate system size fairly accurately.

6. BACKENDS

The network, measurement, and control interfaces abstract the experimental environment in which protocols are tested. In order to use an alternative platform with an experiment written on the AgentScope interfaces an experimenter need only write a small set of adaptor classes. A backend adaptor maps the abstract concepts from the AgentScope interfaces onto concrete implementa-

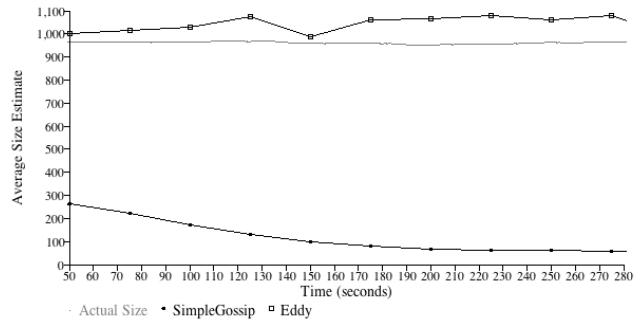


Figure 2: Size estimates: 1000 agents with churn.

tions within a platform. While backends all provide the same basic functionality, they may vary significantly in the performance guarantees they provide for those functions. For instance a simulator may guarantee that messages will always be delivered with minimal delay, while in an emulation environment message and agent failures may be common occurrences. Different environments also include different support functions. Single machine environments can easily provide support for precise measurements of agent behaviour. In distributed environments measurement can require expensive coordination and communication, and thus may necessarily be less detailed. It is therefore natural that as a protocol proceeds through the development cycle the most appropriate environment for testing it will change.

In the example experiment the Platform 9 3/4 simulator can be replaced by the AgentScope agent middleware [15] simply by changing which `Environment` is created in line 65. Figure 3 compares the size estimate distributions for the Eddy protocol on 50 agents running in AgentScope and on Platform 9 3/4. Each backend has its advantages and disadvantages. Since AgentScope is a full agent operating system, testing is not limited to aspects that were designed into the environmental model. Run with the 1 second gossip interval, t , used in the original experiments, the Eddy protocol swamps AgentScope communications. Gossiping protocols are often designed assuming low-cost communication methods, such as UDP [3], while AgentScope communication is designed to be reliable and to maximise security. The “AgentScope Congested” trial shows that messages timing out in Eddy results in an under-estimate of the system size. Slowing the gossip rate to 10 seconds shows that Eddy performs roughly equally both in simulation and in the real environment, indicating that it does not have any inadvertent dependencies on shared data or the synchronised timing of the simulation environment. AgentScope, however, is limited to testing smaller numbers of agents than the simulator since each AgentScope agent runs as one or more threads, nor can it easily be setup to mimic churn. Run on AgentScope, the deficiencies of the SimpleGossip protocol seen in Figures 1 and 2 would not be as apparent.

7. DISCUSSION

The design of AgentScope centres on increasing the potential impact of experimental work. The example in Section 5 demonstrates the following improvements over an ad-hoc platform-specific approach to experimental design:

- *Explicit services:* Rather than viewing services as being provided by the backend platform, services (protocols) are moved above the experimental interface. This ensures that the full cost and complexity of an agent algorithm is clearly visible. It also widens the range of platforms and the range of agent

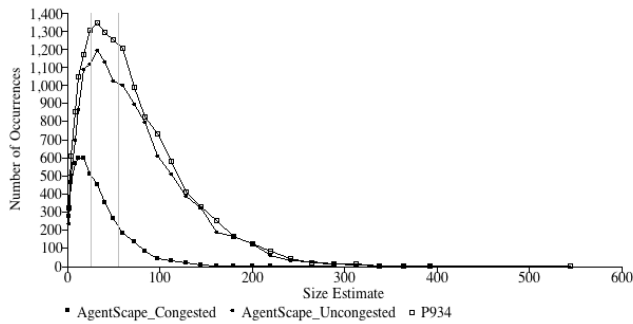


Figure 3: Size estimates: 50 agents on AgentScope.

models that can be supported. Finally, it allows AgentScope to be limited to a small simple interface that is easier to adopt and integrate with existing work.

- *Parity of protocol, scenario, and measurement:* By giving the interfaces for protocol development, experiment development and measurement equal weight, the need for reusability of all three is emphasised. An agent or protocol designer need not start from scratch, but can use existing services, scenarios and metrics. This allows researchers to focus on developing and testing the novel aspect of their ideas.
- *Abstraction of platform type:* Experiments are not specific to one type platform. The reusability of code over different stages in the development cycle is improved, and direct comparisons of performance on different platforms can be made. This has implications for the backends used, each platform need no longer provide a full solution. Timing is an important example, rather than requiring that distributed platforms include the ability to do tests with synchronised time or to create system snapshots, an approach can be used where these tests are done on a simpler centralised platform, and other metrics used to confirm that performance does not change when an experiment is run in a distributed setting.

8. CONCLUSIONS AND FUTURE WORK

This paper examines a comprehensive, long-term approach to multi-agent systems development, and in particular the experimental phases of development. A survey of recent experimental work observes a scarcity of experimental work in the later stages of development, along with a limited level of code reuse. AgentScope, a small set of interfaces on which platform generic experiments can be built is presented. The AgentScope interfaces abstract agent communication, the full setup of experimental scenarios and performance measurement. A demonstration experiment shows how AgentScope can support rapid development of flexible and reusable experiments.

AgentScope promotes a view of the multi-agent systems development process as a long-term research effort by a community of researchers, developing ideas from theory to practice. AgentScope’s design centres around improving researchers’ ability to transfer ideas between projects, groups, or institutions. The aim is to enable a comprehensive development process by improving the reusability of all parts of an experiment. Improvements in reusability support the creation of libraries of protocols, scenarios and metrics that can allow researchers to quickly incorporate previous work into their own experiments. With such changes, experiments become more easily reproducible, and new algorithms can be directly compared

to old, making improvements and tradeoffs clear. Published libraries of scenarios, in which practitioners share their experience of expected use-cases, can be created, enabling input from industry experts to be incorporated in academic work.

Future work involves developing libraries of protocols, scenarios and metrics. Current work in this area focuses on the domain of distributed energy resource management, a highly multi-disciplinary field. The aim is to show that sophisticated agent-based experimentation can be made accessible to a broad audience. The Measurement package and Web Interface package for the AgentScope toolkit are also under development.

9. REFERENCES

- [1] *Autonomous Agents and Multi-Agent Systems*, volumes 18-21. Springer, 2009-2010.
- [2] P. S. da Silva and A. C. V. de Melo. Reusing models in multi-agent simulation with software components. In *AAMAS ’08: Proc. 7th Int Joint Conference on Autonomous Agents and Multiagent Systems*, pages 1137–1144, 2008.
- [3] N. Drost, E. Ogston, R. V. van Nieuwpoort, and H. E. Bal. ARR: real-world gossiping. In *HPDC ’07: Proc. 16th Int. Symposium on High Performance Distributed Computing*, pages 147–158, 2007.
- [4] L. Gasser and K. Kakugawa. MACE3J: fast flexible distributed simulation of large, large-grain multi-agent systems. In *AAMAS ’02: Proc. 1st Int. Joint Conference on Autonomous Agents and Multiagent Systems*, pages 745–752, 2002.
- [5] J. Himmelspach, M. Rohl, and A. M. Uhrmacher. Component-based models and simulations for supporting valid multi-agent system simulations. *Applied Artificial Intelligence*, 24:414–442, 2010.
- [6] B. Horling, R. Mailler, and V. Lesser. Farm: A Scalable Environment for Multi-Agent Development and Evaluation. In A. G. C. Lucena, J. C. A. Romanovsky, and P. Alencar, editors, *Advances in Software Engineering for Multi-Agent Systems*, pages 220–237. Springer-Verlag, 2004.
- [7] H. Kitano and S. Tadokoro. Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI Magazine*, 22(1):39–52, 2001.
- [8] J. Niu, K. Cai, S. Parsons, P. McBurney, and E. Gerding. What the 2007 TAC market design game tells us about effective auction mechanisms. *Journal of Autonomous Agents and Multi-Agent Systems*, 2010.
- [9] I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer server: a tool for research on multi-agent systems. *Applied Artificial Intelligence*, 12:233–250, 1997.
- [10] M. J. North, N. T. Collier, and J. R. Vos. Experiences creating three implementations of the repast agent modeling toolkit. *ACM Trans. Model. Comput. Simul.*, 16(1):1–25, 2006.
- [11] M. Oey, S. van Splunter, E. Ogston, M. Warnier, and F. Brazier. A framework for developing agent-based distributed applications. In *WI-IAT ’10: Proc. IEEE/WIC/ACM Int. Joint Conference on Web Intelligence and Intelligent Agent Technology*, pages 470–474, 2010.
- [12] E. Ogston and S. Jarvis. Peer sampling with improved accuracy. *Peer-to-peer Networking and Applications*, 2(1):51–71, 2009.
- [13] R. Vincent, B. Horling, and V. R. Lesser. An agent infrastructure to build and evaluate multi-agent systems: The java agent framework and multi-agent system simulator. In *Revised Papers from the International Workshop on Infrastructure for Multi-Agent Systems*, pages 102–127. Springer-Verlag, 2001.
- [14] B. White, J. Lepreau, L. Stoller, R. Ricci, S. Guruprasad, M. Newbold, M. Hibler, C. Barb, and A. Joglekar. An integrated experimental environment for distributed systems and networks. *SIGOPS Oper. Syst. Rev.*, 36(SI):255–270, 2002.
- [15] N. J. E. Wijngaards, B. J. Overinder, M. van Steen, and F. M. T. Brazier. Supporting internet-scale multi-agent systems. *Data and Knowledge Engineering*, 41(2-3):229–245, 2002.
- [16] M. Wooldridge, N. R. Jennings, and D. Kinny. The gaia methodology for agent-oriented analysis and design. *Journal of Autonomous Agents and Multi-Agent Systems*, 3:285–312, 2000.

Agent programming with priorities and deadlines

Konstantin Vikhorev Natasha Alechina Brian Logan

School of Computer Science
University of Nottingham
Nottingham NG8 1BB, UK
{kxv,nza,bsl}@cs.nott.ac.uk

ABSTRACT

We present AgentSpeak(RT), a real-time BDI agent programming language based on AgentSpeak(L). AgentSpeak(RT) extends AgentSpeak intentions with *deadlines* which specify the time by which the agent should respond to an event, and *priorities* which specify the relative importance of responding to a particular event. The AgentSpeak(RT) interpreter commits to a priority-maximal set of intentions: a set of intentions which is maximally feasible while preferring higher priority intentions. We prove some properties of the language, such as guaranteed reactivity delay of the AgentSpeak(RT) interpreter and probabilistic guarantees of successful execution of intentions by their deadlines.

Categories and Subject Descriptors

I.2 [Artificial Intelligence]: Programming Languages and Software

General Terms

Languages, Theory

Keywords

Agent programming languages, Belief Desire and Intention logics, Complexity of reasoning

1. INTRODUCTION

Belief-Desire-Intention (BDI) based agent programming languages facilitate the development of rational agents specified in terms of beliefs, goals and plans. They allow an agent to balance deliberating about which plan to adopt in response to events (changes in its beliefs or goals) and executing its current intentions. An agent is *rational* if it adopts and executes intentions which achieve its goals, given its current beliefs.

If an agent's task environment is *real-time*, the requirements for rational behaviour are more complex. In a real-time environment, the events to which the agent must respond are characterized by a *deadline*, e.g., the time by which a goal must be achieved or the agent must respond to a change in its beliefs. In such an environment, a rational agent should not adopt an intention which it believes cannot be successfully executed by its deadline or continue to execute an intention after its deadline. For example, an agent

should not adopt an intention of writing a research proposal which must be submitted by 4pm on Friday if there is insufficient time to write the proposal. Similarly, if the agent believes it cannot achieve every goal or respond to every change in its environment by the relevant deadline, it should adopt intentions for the highest priority events which are feasible.

We define a *real-time BDI agent* as one which is rational in this sense, i.e., it adopts and schedules intentions so as to respond to events by their deadlines; if not all events can be processed by their deadlines, the agent favours intentions responding to high priority events. For a real-time BDI agent, correctness of the agent's program depends not only on the actions the agent performs but the time at which it performs them. However programming real-time BDI agents in most existing BDI languages is hard as they lack the notion of a deadline or the ability to deliberate about the feasibility of intentions.

In this paper we present AgentSpeak(RT), a programming language for real-time BDI agents in applications such as UAVs, process control, trading agents, etc. AgentSpeak(RT) extends AgentSpeak(L) [9] with deadlines and priorities, and, given the estimated execution time of plans, schedules intentions so as to achieve a priority-maximal set of intentions by their deadlines with a specified level of confidence. Real-time tasks can be freely mixed with tasks for which no deadline and/or priority has been specified, and if no deadlines and priorities are specified, the behaviour of the agent defaults to that of a non real-time BDI agent. We prove a number of properties of AgentSpeak(RT), including that the reactivity delay of an AgentSpeak(RT) agent is bounded, that it commits to a priority-maximal set of intentions, and that in a static environment its intentions will complete successfully by their deadlines with specified confidence. We also develop a model of the 'difficulty' of the agent's environment, and show how it can be used to determine the priority of intentions which will complete successfully by their deadlines with specified confidence. A key contribution of the paper is the analysis of real-time guarantees for BDI agents and how these can be achieved within a BDI programming framework. Although our approach to real-time BDI agents is developed in the context of a particular BDI agent programming language, we believe it can be applied to other BDI-based agent programming languages.

The remainder of this paper is organised as follows. In section 2 we present the syntax of AgentSpeak(RT) and briefly describe the execution cycle of the AgentSpeak(RT) architecture. In section 3 we show that under certain reasonable assumptions, the time required to execute a single cycle of the AgentSpeak(RT) interpreter (and hence the reactivity delay of the agent) is bounded. We also prove that an AgentSpeak(RT) agent commits to a priority-maximal set of intentions, and that in a static environment its intentions will

Cite as: Agent programming with priorities and deadlines, Vikhorev, Alechina and Logan, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 397–404. Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

complete successfully by their deadlines with specified confidence. In section 4 we develop a model of the ‘dynamism’ of the agent’s environment, and show how it can be used to determine the priority of intentions which can be reliably scheduled, and to estimate the probability that a scheduled intention of given priority will be displaced from the schedule by the arrival of an intention of higher priority. Finally, in sections 5 and 6 we briefly review related work and conclude.

2. THE AgentSpeak(RT) ARCHITECTURE

In this section we introduce the AgentSpeak(RT) agent programming language and its associated interpreter.

We assume that an AgentSpeak(RT) agent operates in a real-time task environment in which events (external goals and changes in the agent’s beliefs about its environment) may be associated with a deadline and/or a priority. The agent responds to events by adopting and executing intentions. A developer can specify a required level of confidence for the successful execution of intentions in terms of a probability, α , and the agent schedules its intentions so as to ensure that the probability that intentions complete by their deadlines is at least α .¹ Setting $\alpha = 1$ gives hard real-time behaviour, i.e., the agent will only commit to intentions that are certain to complete by their deadlines. If not all intentions can be executed with the required level of confidence, the agent favours intentions triggered by high priority events.

The syntax and semantics of AgentSpeak(RT) is based on AgentSpeak(L) [9]. We briefly review the syntax of AgentSpeak(L) and explain the extensions to support real-time applications. To illustrate the syntax of AgentSpeak(RT) we use a simple running example of a trading agent which buys commodities in an electronic marketplace. The agent receives requests from clients to bid on their behalf, and notifications of goods for sale which the agent may also bid for on its own behalf. The deadline of an event is the deadline for the corresponding auction. The priority is determined by the importance of the client and the type of goods for sale.

The AgentSpeak(RT) architecture consists of five main components: a belief base, a set of events, a plan library, an intention structure, and an interpreter.

2.1 Beliefs and Goals

The agent’s beliefs represent its information about its environment, e.g., sensory input, information about other agents, etc. Beliefs are represented as ground atomic formulas. For example, the agent may believe that client1 is a client, and that good1 is the type of good that the agent buys:

```
client(client1)
buys(good1)
```

A belief atom or its negation is referred to as a *belief literal*. A ground belief atom is called a *base belief*, and the agent’s belief base is a conjunction of base beliefs.

A goal is a state the agent wishes to bring about or a query to be evaluated. An achievement goal, written $!g(t_1, \dots, t_n)$ where t_i, \dots, t_n are terms, specifies that the agent wishes to achieve a state in which $g(t_1, \dots, t_n)$ is a true belief. A test goal, written $?g(t_1, \dots, t_n)$, specifies that the agent wishes to determine if $g(t_1, \dots, t_n)$ is a true belief. For example, the goals

```
!bid(client1, a101, price1)
```

¹For simplicity, we assume that α is the same for all intentions; however the real time guarantees we prove in sections 3 and 4 still hold if α is different for different events.

```
?credit(client1, price1)
```

indicate that the agent should bid price1 in auction a101 on behalf of client1, and determine if client1 has sufficient credit to cover price1.

2.2 Events

Changes in the agent’s beliefs or the acquisition of new achievement goals give rise to events. An addition event, denoted by $+$, indicates the addition of a base belief or an achievement goal. A deletion event, denoted by $-$, indicates the retraction of a base belief.² We distinguish between internal and external events. An external event is one originating in the agent’s environment while internal events result from the execution of the agent’s program. As in AgentSpeak(L), all belief change events are external, while goal change events may be external (goals originated by a user or another agent) or internal (subgoals generated by the agent’s program in response to an external event).

To allow the specification of real-time tasks, external events may optionally specify a deadline and a priority. A *deadline* specifies the time by which a goal should be achieved or the agent should respond to a change in its beliefs. Deadlines are expressed as real time values in some appropriate units, e.g. a user may specify a deadline for a goal as “4pm on Friday”. Deadlines in AgentSpeak(RT) are hard—it is assumed that there is no value in achieving a goal or responding to a belief change after the deadline has passed. A *priority* specifies the relative importance of achieving the goal or responding to a belief change. Priorities define a partial order over events and are expressed as non-negative integer values, with larger values taken to indicate higher priority. For example, the events

```
+!bid(client2, a102, price2)[1010, 15]
+auction(a201, good1)[1060, 10]
```

indicates the acquisition of a goal to bid price2 on behalf of client2 in auction a102 with deadline 1010 and priority 15, and a new belief that good1 is being offered in auction a201, with deadline 1060 and priority 10. By default the deadline is equal to infinity and the priority is equal to zero.

2.3 Plans

Plans specify sequences of actions and subgoals an agent can use to achieve its goals or respond to changes in its beliefs. The head of a plan consists of a triggering event which specifies the kind of event the plan can be used to respond to, and a belief context which specifies the beliefs that must be true for the plan to be applicable. The body of a plan specifies a sequence of actions and (sub)goals to respond to the triggering event.

Actions are the basic operations an agent can perform to change its environment in order to achieve its goals. Actions are denoted by *action symbols* and are written $a(t_1, \dots, t_n)$ where a is an action symbol and t_1, \dots, t_n are the (ground) arguments to the action. For example, the action

```
send-bid(a102, price2)
```

will cause the agent to bid price2 in auction a102. Performing an action may result in changes in the agent’s beliefs when the action’s effects on the environment are sensed at subsequent cycles of the interpreter.

Plans may also contain achievement and test (sub)goals. Achievement subgoals allow an agent to choose a course of action as part of a larger plan on the basis of its current beliefs. An achievement subgoal $!g(t_1, \dots, t_n)$ gives rise to a internal goal addition event

²In the interests of brevity, we do not consider goal deletion events.

$+!g(t_1, \dots, t_n)$ which may in turn trigger subplans at the next execution cycle. Test goals are evaluated against the agent's belief base, possibly binding variables in the plan. For example, the plan

```
+!bid(C, A, P, ) : client(C) <-
  ?credit(C, P); send-bid(A, P).
```

is triggered by a request from agent C to bid price P in auction A. If C is a client, then the agent will check that the client has sufficient credit and, if so, make the bid on the client's behalf.

The BNF for plans is given below:

```
plan          ::= event [ ":" context ] "<-" body "."
event         ::= "+" [ "!" ] atomic-formula |
               "-" atomic-formula
context       ::= true | literal ( "&" literal )*
literal       ::= atomic-formula | "not" atomic-formula
atomic-formula ::= p(t1, ..., tn)
body          ::= true | step ( ";" step )*
step          ::= a(t1, ..., tn) | "!" atomic-formula |
               "? " atomic-formula
```

where p and a are respectively predicate and action symbols of arity $n \geq 0$, and t_1, \dots, t_n are terms. (As in Prolog, constants are written in lower case and variables in upper case, and all negations must be ground when evaluated.)

2.4 Execution Time Profiles

In order to determine whether a plan can achieve a goal by a deadline with a given level of confidence, each action and plan has an associated *execution time profile* which specifies the probability that the action or plan will terminate successfully as a function of execution time. We assume that plans can be arbitrarily interleaved, and the estimated execution time of a plan is independent of any other plans the agent is currently executing. The expected execution time for an action or plan ϕ at confidence level α is given by $et(\phi, \alpha)$. We assume that execution times increase monotonically with α , i.e., in general, to have higher confidence that a plan will complete successfully, we need to allow more time for the plan to execute. The shape of the execution time profile will typically be influenced by the (assumed) characteristics of the environment in which the agent will operate. For example, the probability of a plan to move to a location terminating successfully within a given time may be lower in environments with many obstacles than in environments with fewer obstacles. Execution time profiles can be derived from an analysis of the agent's actions, plans and environment, or using automated techniques, e.g., stochastic simulation. In the simple case of plans consisting of a sequence of actions, the execution time profile for the plan can be computed from the execution time profiles of its constituent actions. However for plans which contain subgoals, the execution time profile will depend on the relative frequency with which the possible plans for a subgoal are selected in the agent's task environment.

2.5 Intentions

The intention structure contains plans that have been chosen to achieve goals or respond to changes in the agent's beliefs. Plans triggered by changes in beliefs or the acquisition of an external (top-level) achievement goal give rise to new intentions. Plans triggered by the processing of an achievement subgoal in an already intended plan are pushed onto the intention containing the subgoal. Each intention consists of a stack of partially executed plans, a set of substitutions for plan variables, and a deadline and priority. The set of variable substitutions for each plan in an intention results from matching the belief context of the plan and any test goals it contains against the agent's belief base. The deadline and priority

of an intention are determined by the triggering event of the root plan.

2.6 The AgentSpeak(RT) Interpreter

The interpreter is the main component of the agent. It manipulates the agent's belief base, events and intention structure, deliberates about which plan to select in response to belief and goal change events, and schedules and executes intentions.

The agent's state is a tuple $\langle B, E, I \rangle$ consisting of a set of base beliefs B , a set of events E , and an (ordered) set of intentions I . We formalize the execution of the interpreter as a sequence of function applications which compute the new state of the agent and an executed action based on its current state and its inputs at the current cycle

$$\langle (B', E', I'), a \rangle = exec(sched(opt(evt(\langle B, E, I \rangle, P, G))))$$

where P is a set of percepts, G is a set of external goal addition events, B', E', I' are the updated belief, event and intention sets, and a is an action or null.

The function evt generates a set of events based on the agent's percepts P and external goal addition events G . It updates the belief base B with the percepts in P to give an updated belief base B' and a set of belief addition and removal events E_P , and returns a new state

$$\langle B', E_1 = E \cup E_P \cup G, I \rangle = evt(\langle B, E, I \rangle, P, G)$$

The second function, opt , takes $\langle B', E_1, I \rangle$ as input and returns a pair consisting of a new state and a set of applicable plans or options O

$$\langle (B', \emptyset, I_1), O \rangle = opt(\langle B', E_1, I \rangle)$$

In contrast to AgentSpeak(L) which processes a single event at each interpreter cycle, to ensure reactivity, AgentSpeak(RT) iterates through E_1 , and, for each event $e \in E_1$, generates a set of applicable plans O_e . A plan is *relevant* if its triggering event can be unified with e and a relevant plan is *applicable* if its belief context is true in B' . In general, there may be many applicable plans or options for each event. A selection function S_O chooses one of these plans for each event to give a set of options $O = \{S_O(O_e) \mid e \in E_1\}$. S_O is a partial function, i.e., it is not defined if O_e is empty. If the event was triggered by a subgoal of an existing intention, failure to find an applicable plan for the subgoal, i.e., if $O_e = \emptyset$, aborts the intention which posted the subgoal and the intention is removed from I (hence the change from I to I_1).

The third function, $sched$, takes $\langle (B', \emptyset, I_1), O \rangle$ as input and returns a new state

$$\langle B', \emptyset, I_2 \rangle = sched(\langle (B', \emptyset, I_1), O \rangle)$$

For each plan π in O , it either pushes π on top of the existing intention in I_1 that generated the triggering event (if the triggering event for π was internal), or creates a new intention τ and adds it to a set I_E (if the triggering event for π was external). I_2 is the result of applying the scheduling algorithm (see Algorithm 2 below) to $I_1 \cup I_E$. The scheduling algorithm returns a set of feasible intentions in deadline order (earliest deadline first).

Finally, $exec$ takes $\langle B', \emptyset, I_2 \rangle$ as input and returns a pair consisting of a new state and an executed action

$$\langle (B', E', I'), a \rangle = exec(\langle B', \emptyset, I_2 \rangle)$$

where I' is the result of executing the first intention in the schedule I_2 , E' contains any internal goal addition event generated by executing the intention, and a is the action executed (or null if no action was executed). Executing an intention involves executing

the first goal or action of the body of the topmost plan in the stack of partially executed plans which forms the intention. Executing an achievement goal adds a corresponding internal goal addition event to E' . Executing a test goal involves finding a unifying substitution for the goal and the agent's base beliefs. If a substitution is found, the test goal is removed from the body of the plan and the substitution is applied to the plan. If no such substitution exists, the test goal is not removed and may be retried at the next cycle. Executing an action results in the invocation of the Java code that implements the action. If the action completes within its expected execution time $et(a, \alpha)$, it is removed from the body of the plan. Actions which time out are not removed and may be retried at the next cycle.³ The executed action is returned as a . Reaching the end of a plan (denoted by *true* below) causes the plan to be popped from the intention and any substitutions for variables appearing in the head of the popped plan are applied to the topmost plan in the intention.

Algorithm 1 AgentSpeak(RT) Interpreter Cycle

```

E := E ∪ G ∪ belief-events(B, P)
B := update-beliefs(B, P)
for all  $\langle e, \tau \rangle \in E$  do
   $O_e := \{\pi\theta \mid \theta \text{ is an applicable unifier for } e \text{ and plan } \pi\}$ 
   $\pi\theta := S_O(O_e)$ 
  if  $\pi\theta \neq \emptyset$  and  $\tau \notin I$  then
     $I := I \cup \pi\theta$ 
  else if  $\pi\theta \neq \emptyset$  and  $\tau \in I$  then
     $I := (I \setminus \tau) \cup \text{push}(\pi\theta\sigma, \tau)$  where  $\sigma$  is an mgu for  $\pi\theta$  and  $\tau$ 
  else if  $\pi\theta = \emptyset$  and  $\tau \in I$  then
     $I := I \setminus \tau$ 
  end if
end for
I := SCHEDULE(I)
if  $I \neq \emptyset$  then
   $\tau := \text{first}(I)$ 
  if  $\text{first}(\text{body}(\text{top}(\tau))) = \text{true}$  then
     $\pi := \text{pop}(\tau)$ ,  $\pi' := \text{pop}(\tau)$ 
     $\text{push}(\text{head}(\pi') \leftarrow \text{rest}(\text{body}(\pi'))\theta, \tau)$ 
    where  $\theta$  is an mgu such that  $\text{head}(\pi)\theta = \pi'\theta$ 
  else if  $\text{first}(\text{body}(\text{top}(\tau))) = !g(t_1, \dots, t_n)$  then
     $E = \{(+!g(t_1, \dots, t_n), \tau)\}$ 
  else if  $\text{first}(\text{body}(\text{top}(\tau))) = ?g(t_1, \dots, t_n)$  then
    if  $?g(t_1, \dots, t_n)\theta$  is an answer substitution then
       $\pi := \text{pop}(\tau)$ 
       $\text{push}(\text{head}(\pi) \leftarrow \text{rest}(\text{body}(\pi))\theta, \tau)$ 
    end if
  else if  $\text{first}(\text{body}(\text{top}(\tau))) = a(t_1, \dots, t_n)$  then
    if  $\text{execute}(a(t_1, \dots, t_n), et(a(t_1, \dots, t_n), \alpha))$  then
       $\pi := \text{pop}(\tau)$ 
       $\text{push}(\text{head}(\pi) \leftarrow \text{rest}(\text{body}(\pi)), \tau)$ 
    end if
  end if
end if

```

The interpreter code is shown in Algorithm 1. The functions *head* and *body* return the head and body of an intended plan, and *first* and *rest* are used to return the first and all but the first elements of a sequence. The function *top* returns the topmost plan in an intention. The function *pop* removes and returns the topmost plan of an intention and the function *push* takes a plan (and any substitution) and an intention and pushes the plan onto the top of the intention. The function *execute* takes an action and an expected execution time, and executes the action for at most the expected execution time. It returns true if the action completes successfully

³Allowing test goals and actions to be retried is not critical, but means that successful execution of intentions is less dependent on precise characterization of the execution time profile of actions.

within its expected execution time; otherwise it returns false.

The scheduling algorithm is shown in Algorithm 2. The set of candidate intentions is processed in descending order of priority.⁴ A candidate intention is added to the schedule if it can be inserted into the schedule in deadline order while meeting its own and all currently scheduled deadlines. A set of intentions τ_1, \dots, τ_n is feasible if there exists a schedule where each intention is executed before its deadline. To check whether a schedule exists for a set of intentions ordered earliest deadline first, it suffices to check that for every scheduled intention τ_i

$$\sum_{j \leq i} et(\tau_j, \alpha) - ex(\tau_j) \leq d(\tau_i)$$

where $ex(\tau_j)$ is the time τ_j has spent executing up to this point, and $d(\tau_i)$ is the deadline for τ_i . That is, the sum of expected remaining execution time of intentions scheduled earlier than τ_i including τ_i itself is less than the deadline of τ_i . A set of tasks is feasible iff they can be scheduled earliest deadline first [7]. Intentions which are not feasible in the context of the current schedule or which have exceeded their expected execution time are dropped.⁵

Algorithm 2 AgentSpeak(RT) Scheduling Algorithm

```

function SCHEDULE(I)
   $\Gamma := \emptyset$ 
  for all  $\tau \in I$  in descending order of priority do
    if  $\{\tau\} \cup \Gamma$  is feasible then
       $\Gamma := \{\tau\} \cup \Gamma$ 
    end if
  end for
  sort  $\Gamma$  in order of increasing deadline
  return  $\Gamma$ 
end function

```

The scheduler returns a set of intentions which is 'maximally feasible' (no more intentions can be added to the schedule if the scheduled intentions are to remain feasible at the specified confidence level) and moreover, intentions which are dropped are incompatible with some scheduled higher priority intention(s). Scheduling in AgentSpeak(RT) is pre-emptive in that the adoption of a new high-priority intention τ_i may prevent previously scheduled intentions with priority lower than i (including the currently executing intention) being added to the new schedule.

Note that if deadlines and priorities are not specified for external events (and hence $d = \infty, p = 0$ for all intentions), $et(\phi, \alpha) = \infty$ for all $\phi, 0 \leq \alpha \leq 1$, the behaviour of an AgentSpeak(RT) agent defaults to that of a non real-time BDI agent.

2.7 Implementation

We have implemented AgentSpeak(RT) in Java, and the current prototype implementation includes the core language described above and implementations of some basic actions. Additional user-defined actions can be added using a Java API. AgentSpeak(RT) supports

⁴If there are multiple intentions with the same priority and/or deadline, we assume they are processed in a fixed order.

⁵The real time guarantees we prove in section 3 still hold in some circumstances if intentions that exceed their expected execution time are not dropped, but it complicates the presentation. The basic idea is that an intention τ which has exceeded its expected execution time has its priority reduced to 0. τ will only be scheduled if, after scheduling all higher priority intentions, there is sufficient slack in the schedule to execute at least one step in τ before its deadline. Given sufficient slack in the schedule, τ can therefore still complete successfully. It will be however dropped if it exceeds its deadline.

two mechanisms for defining primitive actions: writing a class which implements the `ExternalAction` interface, and direct invocation of methods in existing Java legacy code.

2.8 Example

In this section we sketch a simple example `AgentSpeak(RT)` agent and show how it allows the specification of tasks with priorities and deadlines.

Consider a trading agent which buys commodities in an electronic marketplace both on its own behalf and as a broker on behalf of clients. The market operates as a series of concurrent first-price sealed-bid auctions of short duration in which sellers offer goods for sale. Each auction has a deadline by which bids must be received. Once the deadline for an auction has passed, the market determines the highest bid and notifies successful agents of their purchase and remaining credit level. The trading agent responds to two kinds of events: requests from clients to make a specified bid on their behalf in a particular auction, and notifications of new auctions where the agent may decide to bid on its own behalf. The deadline of an event is the deadline for bids for the corresponding auction, and priorities are assigned to events based on the importance of the client (for client requests) and the type good sold in the auction (for auction notifications). The agent's primary role is as a broker, so the priority of auction notification events is lower than that of client requests. When the agent receives a request to bid in an auction, it checks that the requesting agent is a client and that the client has sufficient credit before making the bid. When it receives notification of a new auction, the agent may decide to bid on its own account. Determining what price it should offer depends on the type of good offered for sale. We require that scheduled intentions complete by their deadlines with probability $\alpha = 0.9$.

The agent's program is shown below.

```
Beliefs:
  client(client1)
  client(client2)
  credit(client1, price1)
  credit(client2, price2)
  credit(agent, price3)
  buys(good1)
  buys(good2)

Plans:
+!bid(C, A, P,) : client(C) <-
  ?credit(C, P); send-bid(A, P).

+auction(A, G) : buys(G) <-
  price(G, P); ?credit(agent, P); send-bid(A, P).
```

At time 1000 the agent receives the following events.

```
+!bid(client1, a101, price1) [1100, 20] A request from
client1 to bid price1 in auction a101. The deadline for this event
is 1050 and the client is important so the priority of this event is 20.
+!bid(client2, a102, price2) [1010, 15] A request from
client2 to bid price2 in auction a102 with deadline 1010 and pri-
ority 15.
+auction(a201, good1) [1060, 10] A notification of an auc-
tion a201 offering good1 with deadline 1060 and priority 10.
```

These events trigger instances of the two plans in the agent's program to give three candidate intentions, τ_1 (plan 1 with triggering event `client1`), τ_2 (plan 1 with triggering event `client2`) and τ_3 (plan 3 with triggering event `a201`). The expected execution time of τ_1 and τ_2 (plan 1) at the specified confidence level $\alpha = 0.9$ is 20. The expected execution time of τ_3 (plan 2), which involves

determining what price the agent should bid, is 50. τ_2 is not feasible, and is dropped. τ_1 and τ_3 are feasible and are scheduled in deadline order: τ_3 is scheduled first from 1000–1050, as it has the earliest deadline, followed by τ_1 from 1050–1070. The agent starts execution of τ_3 .

Consider a new event arriving at time 1030 while τ_3 is still executing (e.g., after the step `price(good1, P)` has been executed but not `?credit(agent, P)`). The new event is a request for the agent to bid in auction `a103` for client2: `+!bid(client2, a103, price2)`. The deadline is 1065 and priority is 15, and the resulting candidate intention, τ_4 has an expected execution time of 20. τ_1 and τ_4 are inserted into the new schedule in deadline order. However it is not possible to schedule τ_3 by its deadline—its expected completion time exceeds its deadline by 20. (Note that it doesn't matter the order in which τ_3 and τ_4 are scheduled, they cannot both be achieved by their deadline.) τ_3 is therefore dropped, and the agent begins to execute τ_4 .

While the plans and events in this example are extremely simple, it illustrates how the agent continually updates its scheduled intentions in response to events to give a priority maximal set of intentions that can be achieved by their deadlines with confidence α .

3. REAL-TIME AGENCY

In this section we show that under certain assumptions (which we believe are reasonable for real-time applications), the time required to execute a single cycle of the `AgentSpeak(RT)` interpreter (and hence the reactivity delay of the agent) is bounded. We also show that an `AgentSpeak(RT)` agent commits to a priority-maximal set of intentions, and that, given a fixed schedule, the probability that an intention will complete successfully by its deadline is α .

We make the following assumptions about the agent's program and task environment:

1. the set of possible beliefs has a fixed maximal size (for example, the set of possible beliefs can be restricted to the set of ground instances of any atomic formula appearing in a belief context or a test goal for a finite set of constants);
2. the set of possible goals has a fixed maximal size (for example, the set of possible goals can be limited to the set of ground instances of any atomic formula appearing in an achievement goal for a finite set of constants);
3. the maximal possible interval between the arrival time and deadline of any event is a constant d_{max} ;
4. the minimal expected execution time for any plan is a constant t_{min} ; and
5. there is a maximal expected execution time, t_{max} , for any action in the agent program (i.e., $t_{max} = \max(et(a, \alpha))$ for any action a at the specified α)

THEOREM 1. *If the sets of possible beliefs and goals, the maximal expected action execution time and the maximal distance to deadline have a fixed maximal size, and the minimal plan execution time has a fixed minimal size, then the time required to execute a single cycle of the `AgentSpeak(RT)` interpreter is bounded by a constant δ_c .*

PROOF. The time required to compute `evt` depends on the size of the sets P and G . If the set of all possible beliefs is limited to a fixed finite set of ground belief atoms (assumption 1 above), then the number of possible percepts $|P|$ is bounded by a constant

(assuming that the agent’s percepts are limited to changes in its beliefs).

If the set of all possible agent goals is similarly limited (assumption 2), then the number of possible goals $|G|$ is also bounded by a constant. This means that $|E|$ and $|B|$ are also bounded by a constant at all stages of the cycle.

The time required to compute opt is bounded if $|B|$ is bounded. Computing the set of applicable plans for each event involves evaluating the belief context of each plan whose trigger matches the event against the agent’s beliefs. Assuming that returning the set of plans which match an event is a constant time operation and matching the belief context of a plan against the agent’s beliefs is bounded by a polynomial in $|B|$, if $|B|$ is bounded, then the time required to compute opt is also bounded by a constant. Computing $sched$ is bounded by a polynomial (in fact, a quadratic function) in $|I|$. In the worst case, when priority varies with deadline and intentions are inserted into the schedule in order of decreasing deadlines, then the feasibility of each new intention involves checking the feasibility of all currently scheduled intentions. $|I|$ is bounded if the maximal possible interval between the arrival time and deadline of any event is a constant d_{max} (assumption 3), and the minimal expected execution time for any plan is a constant t_{min} (assumption 4). Then the maximal possible number of schedulable intentions is bounded by d_{max}/t_{min} . By assumption 5, maximum action execution time and hence the time to compute $exec$ is bounded by a constant t_{max} .

The total cycle execution time is bounded by a constant δ_c which is the sum of the bounds on computing each of the functions. \square

By the *reactivity delay* of an agent we mean the time the system takes to recognize and respond to an external event [4] (i.e., the time from the arrival of the event to the selection of a plan for the event or deciding not to respond to the event).

THEOREM 2. *Given assumptions 1-5, the reactivity delay of an AgentSpeak(RT) agent is bounded.*

PROOF. The maximum reactivity delay is for an event which arrives just after the evaluation of evt begins, which is guaranteed to be responded to by the end of the *next* agent cycle. Since the agent’s cycle is bounded by δ_c , the maximum reactivity delay is hence bounded by $2\delta_c$. \square

Note that the analogous result does not hold for AgentSpeak(L) [9], even when the set of beliefs and goals are bounded. AgentSpeak(L) processes a single event per cycle, and the order in which events are processed is determined by the event selection function S_E . If S_E preferentially returns events of a particular type and events of this type arrive sufficiently frequently, then other events will never be processed.

We now show that an AgentSpeak(RT) agent commits to a priority-maximal set of intentions.

DEFINITION 1. *Consider a set of intentions I . A set $\Gamma \subseteq I$ is a priority-maximal set of intentions (with respect to I) if:*

1. Γ is feasible;
2. $\forall \tau \in I$ such that $\tau \notin \Gamma$: $\{\tau\} \cup \Gamma$ is infeasible;
3. $\forall \tau \in I$ such that $\tau \notin \Gamma$, either $\{\tau\}$ is infeasible, or $\exists \Gamma' \subseteq \Gamma$: the minimal priority of an intention in Γ' is greater or equal to $p(\tau)$, and $\Gamma' \cup \{\tau\}$ is infeasible.

Intuitively, this definition describes a subset of I which is ‘maximally feasible’ (no more intentions from I can be added if the

intentions are to remain feasible at the specified confidence level) and moreover, intentions in $I \setminus \Gamma$ are incompatible with some subset of Γ which contains higher priority intention(s). Observe that if all intentions in I have a unique priority, then there is only one priority-maximal subset of Γ , containing the maximal number of highest priority intentions which are jointly feasible.⁶

THEOREM 3. *Given a partially ordered set of intentions $I = \{\tau_1, \tau_2, \dots, \tau_n\}$, where $p(\tau_i) \geq p(\tau_j)$ for $i < j$, the AgentSpeak(RT) scheduling algorithm generates a priority-maximal set of intentions $\Gamma \subseteq I$.*

PROOF. Note that the AgentSpeak(RT) scheduling algorithm generates a sequence of sets starting with $\Gamma_0 = \emptyset$, and sets Γ_i to be $\Gamma_{i-1} \cup \{\tau_i\}$, $\tau_i \in I$ if $\Gamma_{i-1} \cup \{\tau_i\}$ is feasible in deadline order, or Γ_{i-1} otherwise. The last set Γ_n is Γ . By construction, Γ is a feasible set of intentions. Γ is also clearly a maximally feasible subset of I : there is no $\tau \in I$ such that $\tau \notin \Gamma$ and $\Gamma \cup \{\tau\}$ is feasible. To prove that it is priority-maximal, let $\tau_i \in I$, $\{\tau_i\}$ feasible, and $\tau_i \notin \Gamma$. We need to show that τ_i is incompatible with some subset of Γ which contains only intentions of the same or higher priority than $p(\tau_i)$. Since the intentions are added to Γ in descending order of priority, when τ_i is considered and found incompatible with Γ_{i-1} , $p(\tau_i) \leq \min(\{p(\tau') : \tau' \in \Gamma_{i-1}\})$. \square

THEOREM 4. *The probability that an intention τ will execute successfully in a static environment is equal to α .*

PROOF. Immediate, from the fact that the execution time profiles of plans give us the estimate of duration of the task with the probability α . \square

4. DYNAMIC ENVIRONMENTS

The guarantees in the previous section are for a static environment and schedule. They do not consider cases where a new intention generated by the arrival of an event cannot be scheduled, or a scheduled intention is dropped as a result of the arrival of a higher priority task. In this section we develop a simple model of task arrival which can be used to characterise the ‘difficulty’ of an agent’s task environment. We show how this model can be used to determine the priority of intentions which can be reliably scheduled in an environment of specified difficulty, and to estimate the probability that an intention of given priority will not be displaced from the schedule by the arrival of an intention of higher priority.

We characterise the agent’s task environment in terms of the average arrival rate and time available for the execution of intentions of a given priority. Let r_i be the average triggering rate of intentions of priority i (expressed as the number of triggering events / unit time), and a_i the average time available for their execution, i.e., the difference between the intention’s deadline and the time at which it was triggered. For example, if each external achievement goal has a distinct priority level, r_i and a_i correspond to the arrival rate and average time to achieve a particular type of goal. We assume that $a_i \geq t_i$ where t_i is the average execution time of intentions of priority i at the specified confidence level α , i.e., that deadlines advance with the time at which intentions are triggered

⁶In general, a priority-maximal set of intentions is not guaranteed to contain the largest number of high priority intentions. For example, if $S = \{\tau_1, \tau_2, \tau_3, \tau_4\}$, where $p(\tau_1) = p(\tau_2) = p(\tau_3) = 2$, $p(\tau_4) = 1$, and it is possible to schedule either τ_1 and τ_4 together, or τ_2 and τ_3 together, both sets $\{\tau_1, \tau_4\}$ and $\{\tau_2, \tau_3\}$ will be priority-maximal sets (but, for example, $\{\tau_1\}$ will not be). Computing the set containing the largest number of highest priority intentions is a hard combinatorial problem, which can not be solved by a real-time scheduler.

such that intentions are always individually feasible on average. t_i can be computed from the execution time profiles of the plans in the agent's plan library for the task environment. The larger r_i and the smaller the difference between a_i and t_i , the more difficult the agent's environment. The larger the value of r_i the greater the number of intentions the agent must execute in a given period of time; the smaller the value of $a_i - t_i$ the less time there is to accommodate intentions of priority less than i . In general, the probability that an intention of priority j will be unschedulable is an increasing function of r_i and decreases with $a_i - t_i$ for all $i > j$.

In the worst case, when the schedule is full and intentions complete their execution just before their deadlines, the long term average number of intentions of priority i in the agent's schedule is given by $\lambda_i = r_i a_i$. The amount of uncommitted or 'slack' time unused by intentions of priority i in such a schedule is $s_i = \lambda_i(a_i - t_i)$. We assume that $s_i \geq 0$ for all i given an otherwise empty schedule, i.e., that the average arrival rate and time available for execution of intentions of each priority level are feasible for the agent. For intentions of priority $i - 1$ to be reliably scheduled, the total time required for their execution, $\lambda_{i-1} t_{i-1}$, must be less than s_i . If the maximum priority of any intention is m , then the time available to schedule intentions of priority j is

$$s_{j+1} = \lambda_m(a_m - t_m) - \sum_{i=j+1}^m \lambda_i t_i$$

Hence intentions of priority $j < m$ are typically unschedulable if $s_{j+1} \ll \lambda_j t_j$. For given values of r_i , a_i and t_i , we can therefore determine the priorities of intentions which can be typically scheduled.

For a new intention of priority j to be schedulable, there must be at least t_j slack in the schedule at level j , i.e., $s_j \geq t_j$. The amount of slack at priority level j in the schedule depends on the number of intentions in the schedule at priority levels j, \dots, m . (A new intention of priority j can displace already scheduled intentions with priority $< j$ but not already scheduled intentions of priority j or higher.) Any currently scheduled intentions of priority i , $j \leq i \leq m$, must have arrived in the last a_i time units, i.e., between $-a_i$ and now. The number of intentions of each priority level j, \dots, m arriving between times $-a_j, \dots, -a_m$ and now can be represented as a vector $\langle f_j, \dots, f_m \rangle$ where f_i for $i \in \{j, \dots, m\}$ is the number of intentions of priority i which arrive within the last $-a_i$. Thus, for an intention of priority j to be schedulable, the following must hold:

$$t_j \leq f_m(a_m - t_m) - \sum_{i=j}^m f_i t_i$$

Let the set of vectors satisfying this condition be

$$F = \{ \langle f_j, \dots, f_m \rangle : f_m(a_m - t_m) - \sum_{i=j}^m f_i t_i \geq t_j \}$$

The probability that an intention of priority j is schedulable, F_j , is then the probability that at most the number of intentions of each priority level specified by one such sequence of arrivals occurs. If the arrival of triggering events is a Poisson process, the probability that at most f_i intentions are added to the schedule in time a_i is given by

$$F(f_i) = \sum_{x=0}^{f_i} \frac{e^{-\lambda_i} \lambda_i^x}{x!}$$

That is, the probability that exactly 0 or 1 or 2 or \dots or f_i intentions are added to the schedule in an interval of length a_i . The probability

that at most the number of intentions of each priority level specified by one such sequence occurs is then

$$F_j = 1 - \prod_{\langle f_j, \dots, f_m \rangle \in F} (1 - F(f_j) \times \dots \times F(f_m))$$

We can also determine the probability that a scheduled intention of priority j is displaced from the schedule by the arrival of a higher priority intention. If the uncommitted time at priority m , s_m , is sufficient to schedule the expected number of intentions of priority $m - 1$, then for an intention of priority $m - 1$ to be displaced from the schedule, $u_m = \lceil s_{m-1}/t_m \rceil$ intentions must be added to the schedule during time a_{m-1} . The expected number of priority m intentions arriving in time a_{m-1} is $\lambda_m = r_m a_{m-1}$. The probability that at least u_m intentions are added to the schedule in time a_{m-1} is given by

$$U(u_m) = 1 - F(u_{m-1})$$

That is, $1 -$ the probability that exactly 0 or 1 or 2 or \dots or u_{m-1} events arrive in an interval of length a_{m-1} .

In general, for a scheduled intention of priority $j < m$ to be displaced, sufficient intentions of priority $> j$, with total execution time $> s_j$, must arrive within a time interval a_j . A set of intentions with priorities $j + 1, j + 2, \dots, m$ sufficient to displace an intention of priority j can be represented as a vector $\langle u_{j+1}, \dots, u_m \rangle$ where $u_i \in \{j + 1, \dots, m\}$ is the number of intentions of priority i which arrive within a_j . To displace an intention of priority j such vectors must satisfy a number of conditions. First, the number of intentions of each priority must be feasible given s_j . Second, the combined execution time of all intentions in the set must be greater than s_j . Third, that the combined execution time of the intentions should exceed s_j by at most the least execution time of any intention in the set. Let the set of vectors satisfying the conditions be

$$U = \{ \langle u_{j+1}, \dots, u_m \rangle : \begin{aligned} 0 &\leq u_i \leq s_j/t_i, \\ \sum u_i t_i &> s_j, \\ \sum u_i t_i - \min_{i \in \{j+1, \dots, m\}}(t_i) &\leq s_j \end{aligned} \}$$

That is, the set of all possible sequences of intentions of priority $> j$ which have combined execution time "just greater" than s_j . The probability that an intention of priority j is displaced, U_j , is then the probability that at least the number of intentions of each priority level specified by one such sequence occurs

$$U_j = 1 - \prod_{\langle u_{j+1}, \dots, u_m \rangle \in U} (1 - (U(u_{j+1}) \times \dots \times U(u_m)))$$

where $U(u_i) = 1 - \sum_{x=0}^{u_i-1} \frac{e^{-\lambda_i} \lambda_i^x}{x!}$ as above.

The probability that an AgentSpeak(RT) agent will execute an intention of priority j to completion is then

$$E_j = F_j \times (1 - U_j) \times \alpha$$

For given values of r_i , a_i and t_i , we can therefore determine the probability that an intention of given priority will not be scheduled, or will be displaced from the schedule by a higher priority intention before it can complete successfully. For example, given the rate at which requests to bid arrive and the time it takes to execute the agent's plan to process bids, we can determine the probability that an intention to bid will be executed. For different applications and priority levels, different probabilities of execution may be appropriate. If, for the intended application, E_j is deemed to be too low, the agent developer must either reduce the average triggering rate of intentions of priority $> j$ or increase $a_i - t_i$ for $i \geq j$, e.g., by reducing the execution time of the agent's plans.

5. RELATED WORK

A number of agent architectures and platforms have been proposed for the development of agents which must operate in highly dynamic environments. For example, the Procedural Reasoning System (PRS) [5] and PRS-like systems, e.g., JAM [6] and SPARK [8], have features such as metalevel reasoning which facilitate the development of agents for real time environments. However, to guarantee real time behaviour, these systems have to be programmed for each particular task environment—there are no general methods or tools which allow the agent developer to specify that a particular goal should be achieved by a specified time or that an action should be performed within a particular interval of an event occurring. In contrast, AgentSpeak(RT) provides a high-level programmatic interface to a standardised real-time reasoning mechanism for tasks with different priorities and deadlines.

Perhaps the work most similar to that described here are architectures such as the Soft Real-Time Agent Architecture [12] and AgentSpeak(XL) [1]. These architectures use the TÆMS (Task Analysis, Environment Modelling, and Simulation) framework [3] together with Design-To-Criteria scheduling [13] to schedule intentions. TÆMS provides a high-level framework for specifying the expected quality, cost and duration of methods (actions) and relationships between tasks (plans). Like AgentSpeak(RT), methods and tasks can have deadlines, and TÆMS assumes the availability of probability distributions over expected execution times (and quality and costs). DTC decides which tasks to perform, how to perform them, and the order in which they should be performed, so as to satisfy hard constraints (e.g., deadlines) and maximise the agent's objective function. In comparison to AgentSpeak(RT), TÆMS allows the specification of more complex interactions between tasks, and DTC can produce schedules which allow interleaved or parallel execution of tasks. However the view of 'real-time' used in these systems is different from that taken by AgentSpeak(RT). Deadlines are not hard (tasks still have value after their deadline) and no attempt is made to offer probabilistic guarantees regarding the successful execution of tasks. In addition, although DTC can be used in an 'anytime' fashion, neither SRTA or AgentSpeak(XL) execute in bounded time. In [11] we described ARTS, a version of PRS which allows the specification of deadlines and priorities, but which does not provide real-time guarantees for the execution of intentions.

6. CONCLUSIONS

The AgentSpeak(RT) architecture provides a flexible framework for the development of real-time BDI agents. An AgentSpeak(RT) agent will achieve a priority-maximal set of intentions by their deadlines with specified confidence. If not all intentions can be achieved by their deadlines, the agent prefers intentions with greater priority. By varying the confidence level, the developer can control the degree of 'optimism' the agent adopts when determining the time required to complete a task in a given environment. In task environments requiring a higher level of confidence, the agent will typically allow more time to complete tasks (and so schedule fewer tasks). As tasks are scheduled in priority order, increasing the level of confidence also has the effect of causing the agent to focus more on high priority tasks at the expense of lower priority tasks which might be achievable given a more optimistic view of execution time. Real-time tasks can be freely mixed with tasks for which no deadline and/or priority is specified. Tasks without deadlines will be processed after any task with a specified deadline. If no deadlines or priorities are specified, the behaviour of the agent defaults to that of a non real-time BDI agent. Although the ap-

proach to real-time BDI agents we have presented here has been developed in the context of a particular BDI agent programming language, we believe it can be applied to other BDI-based agent programming languages.

AgentSpeak(RT) adopts a single-threaded execution model. While this is appropriate for the majority of real-time applications where the execution times of intentions are relatively short, there are situations where it would be advantageous to be able to execute long-running actions and plans in parallel with other intentions. In future work we plan to extend the AgentSpeak(RT) architecture to allow the parallel execution of intentions as in [2, 10] and investigate alternative approaches to handling plan failure.

7. REFERENCES

- [1] R. 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 *Proc. of the 1st Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS'02)*, pages 1294–1302, 2002.
- [2] B. J. Clement, E. H. Durfee, and A. C. Barrett. Abstract reasoning for planning and coordination. *J. of Artificial Intelligence Research*, 28:453–515, 2007.
- [3] K. S. Decker and V. R. Lesser. Quantitative modeling of complex environments. *Int. J. of Intelligent Systems in Accounting, Finance and Management*, 2:215–234, 1993.
- [4] M. P. Georgeff and F. F. Ingrand. Decision-making in an embedded reasoning system. In *Proc. of the 11th Int. Joint Conf. on Artificial Intelligence (IJCAI'89)*, pages 972–978.
- [5] M. P. Georgeff and A. L. Lansky. Procedural knowledge. *Proc. of the IEEE, Special Issue on Knowledge Representation*, 74(10):1383–1398, 1986.
- [6] M. J. Huber. JAM: a BDI-theoretic mobile agent architecture. In *Proc. of the 3rd Annual Conference on Autonomous Agents (AGENTS'99)*, pages 236–243, 1999.
- [7] J. Y.-T. Leung and J. Whitehead. On the complexity of fixed-priority scheduling of periodic, real-time tasks. *Performance Evaluation*, 2(4):237–250, 1982.
- [8] D. Morley and K. Myers. The SPARK agent framework. In *Proc. of the 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'04)*, pages 714–721, 2004.
- [9] A. S. Rao. AgentSpeak(L): BDI agents speak out in a logical computational language. In *Proc. of the 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World: Agents Breaking Away*, pages 42–55, 1996.
- [10] J. Thangarajah and L. Padgham. An empirical evaluation of reasoning about resource conflicts. *Proc. of the 3rd Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS'03)*, 3:1298–1299, 2004.
- [11] K. Vikhorev, N. Alechina, and B. Logan. The ARTS real-time agent architecture. In *Languages, Methodologies, and Development Tools for Multi-Agent Systems*, Vol 6039 of *LNCIS*, pages 1–15. 2010.
- [12] R. Vincent, B. Horling, V. Lesser, and T. Wagner. Implementing soft real-time agent control. In *Proc. of the 5th Int. Conf. on Autonomous Agents (AGENTS'01)*, pages 355–362, 2001.
- [13] T. Wagner, A. Garvey, and V. Lesser. Criteria-directed heuristic task scheduling. *Int. J. of Approximate Reasoning*, 19:91–118, 1998.

Rich Goal Types in Agent Programming

Mehdi Dastani
Utrecht University
The Netherlands
mehdi@cs.uu.nl

M. Birna van Riemsdijk
Delft University of Technology
The Netherlands
m.b.vanriemsdijk@tudelft.nl

Michael Winikoff
University of Otago
New Zealand
michael.winikoff@otago.ac.nz

ABSTRACT

Goals are central to the design and implementation of intelligent software agents. Much of the literature on goals and reasoning about goals in agent programming frameworks only deals with a limited set of goal types, typically achievement goals, and sometimes maintenance goals. In this paper we extend a previously proposed unifying framework for goals with additional richer goal types that are explicitly represented as Linear Temporal Logic (LTL) formulae. We show that these goal types can be modelled as a combination of achieve and maintain goals. This is done by providing an operationalization of these new goal types, and showing that the operationalization generates computation traces that satisfy the temporal formula.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence—*Intelligent agents, languages and structures*; I.2.5 [Artificial Intelligence]: Programming Languages and Software; F.3.3 [Logics and Meaning of Programs]: Studies of Program Constructs; D.3.3 [Programming Languages]: Language Constructs and Features

General Terms

Theory, Languages

Keywords

Agent Programming, Goals, Formal Semantics

1. INTRODUCTION

A widely-accepted approach to designing and programming agents is the *cognitive* approach, where agents are modeled in terms of mental concepts such as beliefs, goals, plans and intentions. Of the various concepts that have been used for cognitive agents, a key concept is *goals*. This is because agents are (by common definition) proactive, and goals are what allow agents to be proactive. It is also noteworthy that (the existence of explicitly represented) goals is one of the clearer differences between (proactive) agents and active objects. Goals have been extensively studied in artificial intelligence and multi-agent systems (e.g. [2, 13, 16, 21]).

Earlier work focused mostly on achievement goals, which represent a desired state that the agent wants to reach. However, in-

creasingly other *goal types* are being studied such as maintenance goals, which represent a state the agent wants to maintain, and perform goals, which represent the goal to execute certain actions (e.g., [4, 7, 8, 11]). However, only considering a small number of goal types can be limiting, since in practical applications there may be goals that cannot be captured well by achievement or maintenance or perform goals.

To make this discussion more concrete, consider a personal assistant agent that manages a user's calendar and tasks. One goal the agent may have is booking a meeting. This would typically be modelled as an achievement goal that aims to bring about a state where all required participants have the meeting in their calendar. However, in practice, diaries change, and we want to ensure that the meeting remains in participants' diaries, and that should a key participant become unable to attend, a new time will be negotiated. This is not captured by an achievement goal. Rather, it is better modelled by a combined "achieve then maintain" goal which achieves a certain condition, and then maintains it over a certain time period. Another task that we might want the agent to undertake is to ensure that booking travel is not done until the budget is approved. Note that budget approval may be under the control (or perhaps just influence) of the agent, i.e. the agent may have plans for attempting to have the budget approved. Alternatively, it may be completely outside the agent's control, in which case the agent can just wait for it to happen and then enable the travel booking process.

A number of papers have taken this line of research a step further by taking arbitrary Linear Temporal Logic (LTL) formulae as goals [1, 2, 12, 13, 16], rather than considering specific goal types. The advantage of this approach is that it does not restrict the goal types that can be used. However, a possible disadvantage is that it requires extensive alterations of a more basic agent programming framework, the practical implications of which are not yet clear.

Temporal logic is also used by $M \quad M$ [10], but it is used directly for agent execution, whereas we use temporal logic as a design framework for specifying goal *types* which are mapped to existing implementations of achieve and maintenance goals. Additionally, $M \quad M$ requires a particular format for its rules: all rules are in one of the three forms: $\mathbf{start} \rightarrow \varphi$, or $\psi \rightarrow \bigcirc\varphi$ or $\psi \rightarrow \Diamond\phi$ where φ is a disjunction of literals, ψ is a conjunction of literals, and ϕ is a positive literal.

In this paper, we propose an approach that is somewhere in between those focusing on a limited set of goal types and those in which arbitrary LTL formulae can be taken as goals. We propose an approach in which goals that are represented by relatively complex LTL formulae are operationalized by *translating* these LTL formulae to more basic achieve and maintain goals. The advantage of this approach is that the goal types can be integrated in existing

Cite as: Rich Goal Types in Agent Programming, M. Dastani, M.B. van Riemsdijk, M. Winikoff, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Yolum, Tumer, Stone and Sonenberg (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 405-412. Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

agent programming frameworks that already have an operationalization of achieve and maintain goals. We illustrate the approach by showing how a number of LTL formulae can be translated to achieve and maintain goals.

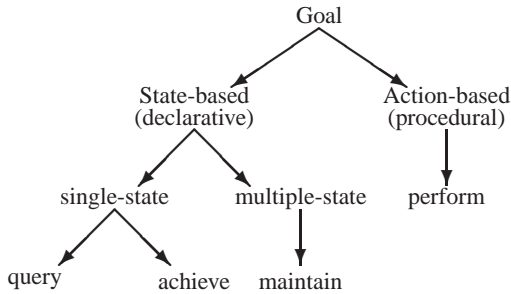
This paper builds on previous work [19] which presented a unifying framework for goals based on viewing goals as LTL formulae that described desired progressions. This previous work captured existing basic goal types, whereas we propose a framework that allows the use of richer goal types. Section 2 provides a brief description of the unifying framework on which we build. Section 3 presents the new goal types which are formalized and realised in sections 4 and 5. Finally, in Section 6 we conclude the paper and discuss some future directions.

2. A UNIFYING FRAMEWORK FOR GOAL TYPES

This paper builds on the framework of van Riemsdijk *et al.* [19] in which a goal type is informally considered as a property characterising a set of computation traces. The framework is explained in terms of an abstract architecture for operationalizing goals such as achieve and maintain goals. In particular, the operationalized architecture aims to capture essential aspects of goals in agent programming frameworks in terms of properties of computation traces, abstracting from particular goal types.

The framework models goals as follows. The state includes state-related propositions, which capture the state of the world, as well as propositions of the form *done_i(a)* which capture the performance of actions. This approach takes an abstract view of a goal type as a particular pattern, or structure, of a formula in Linear Temporal Logic (LTL). A given LTL formula corresponds to a set of traces which make it true, and is viewed as a goal in that it allows a given system trace to be classified as satisfying the goal or not.

van Riemsdijk *et al.* [19] defined four commonly-used goal types: achievement goal, (reactive) maintenance goal¹, perform goal, and query goal. These goals are classified into a taxonomy (below), and an operationalization is provided for them within a single framework, using a simple execution cycle which was extended with additional rules of the form *(condition, action)* where actions could be to S, A or D a goal.



A key feature of this approach is that it balances flexibility with being structured enough to ensure desired properties of goals, and hence for it to make sense for the resulting constructs to be called “goals”. A range of desired properties of goals have been identified in the literature. Winikoff *et al.* [21, Section 2] survey existing literature and, based on this, identify the following desired properties of goals:

1. Persistent: goals should only be dropped for a good reason.

¹Maintenance goals are defined as being *proactive* where violation of desired state is anticipated and avoided, or *reactive*, where the desired state is “recovered” once it is violated [8].

2. Unachieved: goals should not already hold; alternatively, they are dropped when they are achieved.
3. Possible: goals should be dropped when they are impossible to achieve.
4. Known: the agent should be aware of its goals. In implementation terms, this implies a measure of reflectiveness.
5. Consistent: goals should not be in conflict with other adopted goals.

A number of additional properties are identified by [4]. Some of these (Producible/Terminable and Suspendable) simply correspond to the existence of a goal life-cycle. The others (Variable Duration and Action Decoupled) relate to the notion of long term goals that they argue for.

One advantage of the proposed goals framework is that the properties representing a specific goal type can be dealt with in a general and systematic way. Suppose we have a goal ϕ of a certain type. The framework maps ϕ to a goal construct $g(R, \dots)$ where R is a collection of rules (derived from ϕ) that govern transitions between goal states. We can then show that the goal’s realization meets the properties of being persistent, unachieved, and possible, by requiring that the operationalization of $g(R, \dots)$ results in the goal being dropped exactly when ϕ becomes known to be true, or known to be impossible. For example, consider the goal to achieve p . This is mapped [19, Section 3.3.1] to $g(R, \dots)$ where R consists of two rules: one to activate the goal when it is adopted, and one to drop the goal when $s \vee f$ becomes true, where s is the “success condition”, i.e. when the goal is succeeded, here $s = p$; and f is a description of a condition under which the goal becomes impossible to achieve. Here f depends on properties of p , but may be simply false, if it always remains possible to achieve p . Then it is straightforward to show that, under the operational semantics for goals defined by [19], the goal $g(R, \dots)$ is dropped exactly when p becomes known to be true, or known to be impossible (properties 1–3, above). The property of being known (property 4) is achieved by having an explicit goal base which allows the agent to reflect on which goals it has. Consistency (property 5) concerns interactions between goals, and is beyond the scope of this paper.

3. NEW GOAL TYPES

In this paper, goals are represented explicitly as specific formulae in Linear Temporal Logic (LTL) [9]. While [19] defined the notion of goal as representing preferred progressions, and informally referred to LTL to explain this, the operationalization itself did not use LTL explicitly in the representation of goals. The LTL formulae that we use to represent goals are defined by the following grammar. In addition to basic propositions (p), and standard propositional connectives, it has the temporal connectives \Diamond (“eventually”), \Box (“always”), and U (“until”). We use standard abbreviations such as $\top \equiv p \vee \neg p$ and $\phi_1 \vee \phi_2 \equiv \neg((\neg\phi_1) \wedge (\neg\phi_2))$. Note that we do not use the next (O) connective because the goals we consider in this paper does not use this operator.

$$\phi ::= p \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \Diamond\phi \mid \Box\phi \mid \phi_1 \text{U} \phi_2$$

The semantics are the usual ones (given below). They are defined over a model \mathcal{M} which is an infinite sequence of states, where each state is a set of propositions that hold in that state. A formula ϕ is true with respect to a model \mathcal{M} and an index i , indicating the

current state. We use \mathcal{M}_i to denote the i th state in \mathcal{M} .

$\mathcal{M}, i \models p$	iff	$p \in \mathcal{M}_i$
$\mathcal{M}, i \models \neg\phi$	iff	$\mathcal{M}, i \not\models \phi$
$\mathcal{M}, i \models \phi_1 \wedge \phi_2$	iff	$\mathcal{M}, i \models \phi_1$ and $\mathcal{M}, i \models \phi_2$
$\mathcal{M}, i \models \diamond\phi$	iff	$\exists k \geq i : \mathcal{M}, k \models \phi$
$\mathcal{M}, i \models \square\phi$	iff	$\forall k \geq i : \mathcal{M}, k \models \phi$
$\mathcal{M}, i \models \phi_1 \cup \phi_2$	iff	$\exists k \geq i : \mathcal{M}, k \models \phi_2$ and $\forall j$ such that $i \leq j < k : \mathcal{M}, j \models \phi_1$

We now extend the taxonomy of [19] with additional goal types, including those discussed in the introduction. Specifically, the personal assistant agent example introduced new goal types. The first, booking a meeting and then maintaining participant availability, can be formalised as follows. Let pa be short for *participantsAvailable*, ms be short for *meetingScheduled*, and mo be short for *meetingOccurs*, then we represent the goal of booking a meeting as the following LTL formula:

$$\diamond(ms \wedge (pa \cup mo))$$

Considering the second goal, not booking travel until the budget has been approved, this can be formalised as²:

$$(\neg bookTravel) \cup budgetApproved$$

Abstracting from the specific goal instances to general goal types, we have defined goal types of the form $\diamond(\phi_1 \wedge (\phi_2 \cup \tau))$ (achieve ϕ_1 and then maintain ϕ_2 until τ), and $\phi \cup \tau$ (maintain ϕ until τ). We now generalise these goal types by considering a range of ways in which a (non-temporal) property ϕ can be required to hold over a number of states.

Consider a multiple-state goal where a (non-temporal) property ϕ is required to hold over a number of states in the trace. The taxonomy in the previous section (from [19]) only supports a single multiple-state goal. However, there are a number of ways in which a goal pattern can apply to a sequence of states. It can apply:

1. to all states: $\square\phi$;
2. at the start of the trace: $\phi \cup \tau$, where τ is a formula that describes the state at which ϕ is no longer required to be true;
3. at the end of the trace: $\diamond(\tau \wedge \square\phi)$, where τ is a ‘‘trigger’’ formula that describes the state at which ϕ begins to be required to be true; or
4. in the middle of the trace: $\diamond(\tau \wedge (\phi \cup \tau'))$, where τ is the starting trigger and τ' the ending trigger; or
5. it can apply to a number of sub-sequences of states: $\square(\tau \rightarrow (\phi \cup \tau'))$, where τ is a trigger that describes a state at which ϕ begins to be required to hold, and τ' describes the states at which ϕ is no longer required to hold.

These cases for multiple-state goal types are summarised in Figure 1. Note that in all cases we require that ϕ hold at all states within the specified region.

Considering the personal assistant example, the first goal $\diamond(ms \wedge (pa \cup mo))$ corresponds to the fourth case above and the second goal corresponds to the second case above. Note that we could also consider a goal where $\neg bookTravel$ must hold on different sequences of states, e.g. that once there is no more money in the budget (nmm) then travel cannot be booked until a (new) budget is approved, formally: $\square(nmm \rightarrow (\neg bookTravel \cup budgetApproved))$.

²This goal would be expected to be used in conjunction with a goal to book travel, $\diamond bookTravel$.

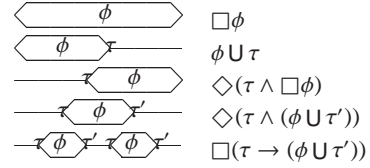


Figure 1: Multiple-state goal types

We thus define the possible LTL patterns that we allow as multiple-state goal types as follows, where ϕ and τ are propositional (i.e. non-temporal). Instead of only supporting a single type of multiple-state goal, as in previous work, we support the following, which correspond with the cases in Figure 1:

$$G_m ::= \square\phi \mid \phi \cup \tau \mid \diamond(\tau \wedge \square\phi) \mid \diamond(\tau \wedge (\phi \cup \tau')) \mid \square(\tau \rightarrow (\phi \cup \tau'))$$

We also allow the single-state goal type $\diamond\phi$. We would like to emphasize that the proposed temporal goal types are by no means exhaustive and that other LTL patterns can be identified to represent other multiple-state goal types as well. Our claim is that the proposed goal types are intuitive in that the corresponding LTL patterns represents desirable execution traces as illustrated by the examples, and that they can be operationalized by means of achieve and maintain goals. We also would like to note that other goal types (e.g., those mentioned in Figure 1) can be represented in our framework as well. For example, the query goal can be represented as $(Bp) \vee (B\neg p)$ (where Bp denotes that agent believes p in the current state), the achieve goal as $\diamond p$, the maintenance goal as $\square p$, and the perform goal as $done(a)$, where $done(a)$ denote the fact that action a is performed.

4. REALISING THE NEW GOAL TYPES

The most characterising feature of our programming approach is to represent goals explicitly as temporal formulae and to operationalize these formulae in terms of achieve and maintain goals. The advantage of this approach is that achieve and maintain goal types have already well-defined operational semantics in some of the existing agent programming frameworks (e.g., 2APL [6], Jadedex [15], and JACK [5]) such that our goal types can be used to extend these frameworks.

In order to realise the new goal types in an operational setting, we assume that an agent configuration comprises a belief base, consisting of propositional atoms, and two goal bases. The first goal base, called the *temporal goal base*, contains goals specified by temporal LTL formulae. The second goal base, called the *basic goal base*, consists of achieve goals of the form $A(\phi)$ (read as: ϕ should be achieved) and maintain goals of the form $M(\phi, \tau)$ (read as: ϕ should be maintained until τ), where ϕ and τ are propositional formulae. The maintain goal $M(\phi, \perp)$ represents that ϕ should be maintained indefinitely.

The operationalization of temporal goal types can then be defined in terms of operations on temporal and basic goal bases. In this paper, we assume that the achieve and maintain goals have a correct operationalization. In particular, for the achieve goal we assume that if $A(\phi)$ is in the basic goal base, then the agent belief base will eventually entail ϕ , and for the maintain goals we assume that if $M(\phi, \tau)$ is in the basic goal base, then the agent belief base entails ϕ until τ is entailed by the belief base. The latter assumption thus ensures that there is no need for reachieving ϕ (typically referred to as reactive maintain goal), since we assume it does not become false once the basic maintain goal has been adopted. The

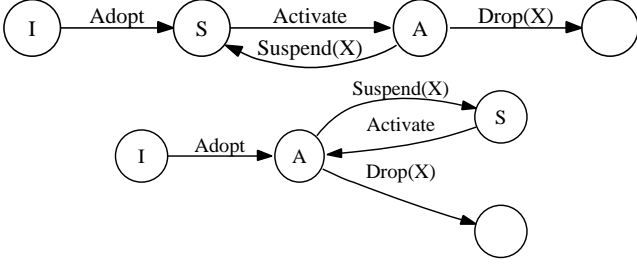


Figure 2: Temporal (top) and basic (bottom) goal life cycles

basic maintain goal is thus interpreted as proactive avoiding the violation of desired states. Clearly, our assumption for the achieve goal is realistic as many agent programming languages such as 2APL [6] provide already programming constructs to implement achieve goals with the correct interpretation. Also, our assumption for the maintain goal is realistic as there exist operationalization proposals for the maintain goals [11] that perform lookahead steps to avoid generating execution paths where the goal is violated.

For the purposes of this paper we want to show that our framework realises temporal goals correctly if we have correct operationalization of achieve and maintain goals. The idea is that the temporal goal $\diamond\phi$ can be operationalised by adding the achieve goal $A(\phi)$ to the basic goal base. Similarly, the temporal goal $\phi \cup \tau$ can be operationalised by adding the maintain goal $M(\phi, \tau)$ to the basic goal base. More complex temporal goals can be operationalized by adding both achieve and maintain goals to the basic goal base, as will be shown in the sequel.

In our framework, goals have a life cycle as illustrated in Figure 2. Both temporal (top) and basic (bottom) goals begin in an initial state (I), and can then be either in active (A) or suspended (S) states. For the reasons explained in the next section, a temporal goal can be adopted entering in a suspended state and a basic goal (either an achieve or a maintain goal) can be adopted entering in the active state. A temporal goal in a suspended state can be activated after which it can be either suspended again or dropped. A basic goal in an active state can be either suspended or dropped.

While temporal goals are assumed to be given by an agent program (specified by an agent programmer), the basic (achieve and maintain) goals are adopted as a consequence of processing temporal goals. In our framework, a temporal goal is dropped if it can be satisfied by adopting an achieve or a maintain goal. A temporal goal is suspended if it can be *partially* satisfied by adopting an achieve or a maintain goal. For this reason, the actions $Drop(X)$ and $Suspend(X)$ drop and suspend the corresponding temporal goals and, at the same time, add the basic goal X to the basic goal base. This notation should not be confused and read as parameterised drop and suspend actions. Finally, the actions $Drop(\emptyset)$ and $Suspend(\emptyset)$ remove a temporal goal from temporal goal base and leave the basic goal base unchanged.

4.1 Life Cycle for Temporal Goals

In order to specify the state transitions of temporal goals, a set of condition-action pairs is assigned to each temporal goal. The condition of such a pair is a test on an agent's belief base and the action is to change the goal's state. The condition-action pairs can be either generic or domain related. A generic condition-action pair specifies a transition for all instances of a goal type while a domain related condition-action pair specifies a transition for a specific instance of a goal type depending on the application at hand.

ϕ	$\delta_g(\phi)$
$\diamond\phi$	$\{\langle\phi, Drop(\emptyset)\rangle, \langle\neg\phi, Drop(A(\phi))\rangle\}$
$\square\phi$	$\{\langle\phi, Drop(M(\phi, \perp))\rangle, \langle\neg\phi, Suspend(A(\phi))\rangle, \langle\phi, Activate\rangle\}$
$\phi \cup \tau$	$\{\langle\phi, Drop(M(\phi, \tau))\rangle, \langle\neg\phi, Suspend(A(\phi))\rangle, \langle\phi, Activate\rangle\}$
$\diamond(\tau \wedge \square\phi)$	$\{\langle\phi \wedge \tau, Drop(M(\phi, \perp))\rangle, \langle\neg(\phi \wedge \tau), Suspend(A(\phi \wedge \tau))\rangle, \langle\phi \wedge \tau, Activate\rangle\}$
$\diamond(\tau \wedge (\phi \cup \tau'))$	$\{\langle\phi \wedge \tau, Drop(M(\phi, \tau'))\rangle, \langle\neg(\phi \wedge \tau), Suspend(A(\phi \wedge \tau))\rangle, \langle\phi \wedge \tau, Activate\rangle\}$
$\square(\tau \rightarrow (\phi \cup \tau'))$	$\{\langle\tau \wedge \phi, Suspend(M(\phi, \tau'))\rangle, \langle\tau \wedge \phi, Activate\rangle\}$

Figure 3: Generic Condition-Action Pairs for Temporal Goals

The domain related condition-action pairs can be used for various purposes, e.g., to activate temporal goals in domain specific situations in which the goals are likely to be realised (in addition to the generic ones) or to suspend them in situations in which the goals are not likely to be realised (in addition to the generic ones). These domain related condition-action pairs should be designed carefully since otherwise they may cause undesirable behavior. We assume domain related condition-action pairs are assigned to each temporal goal by the agent programmer in order to influence the life cycle of temporal goal based on domain dependent knowledge. We use $\delta_g(\phi)$ to refer to the set of generic condition-action pairs of temporal goal ϕ as specified in Figure 3.

It is important to note that these condition-action pairs are only applicable when goals are in specific states, e.g., goals can only be dropped when they are active and activated when they are suspended. The suspension condition-action pairs can not only fire in the active state, but also when a goal is adopted (which moves the goal into the suspended state). The applicability of condition-action pairs are formally specified by the transition rules in Section 5.1.

The first temporal goal type is characterised as $\diamond\phi$, where ϕ is a non-temporal formula. The first generic condition-action pair assigned to this goal indicates that when ϕ is entailed by the agent's beliefs in its current state, then the goal $\diamond\phi$ can be dropped and no basic goal is added to the basic goal base. In this case, the temporal goal is already believed to be satisfied. The second condition-action pair indicates that if ϕ is not entailed by the agent's beliefs in its current state, then the goal $\diamond\phi$ can be dropped, but simultaneously the agent adopts the achieve goal $A(\phi)$ by adding it to its basic goal base. Our assumption that the achieve goal is operationalized correctly ensures that the temporal goal $\diamond\phi$ will be satisfied by the agent execution. As we will see later on, these drop actions only take place if the temporal goal is in its active state. It should be noticed that there is no condition-action pair to activate this temporal goal. We assume such a condition-action pair is added as a domain related pair by the programmer. An example of such a pair is $\langle\tau, Activate\rangle$, which is a strong activation condition as it indicates that the temporal goal should always be activated.

The second temporal goal is characterised by $\square\phi$, where ϕ is a non-temporal formula. The first condition-action pair drops the temporal goal and adopts the maintain goal, adding it at the same time to the basic goal base. Again, our assumption of correct operationalisation of maintain goals ensures that the temporal goal will be satisfied, which is why we can drop the temporal when adopting the basic maintain goal. That is, there is no need to suspend and

reactivate this temporal goal should ϕ no longer be believed, since the latter is assumed not to occur once the basic maintain goal is adopted. Note that a maintain goal $M(\phi, \perp)$ is adopted in a state that satisfies the maintain condition ϕ . The second condition-action pair indicates that the temporal goal $\Box\phi$ should be suspended if ϕ is not entailed by the current agent's beliefs, while adopting the achieve goal $A(\phi)$ to pursue a state from which the maintain goal can be maintained. This can occur only when adopting this temporal goal while ϕ does not hold. Since this temporal goal is dropped after activation, it cannot be suspended again from the active state. The third pair ensures that the temporal goal is activated again upon achievement of ϕ .

The operationalisation of the third temporal goal type is very much similar to the second one since $\Box\phi$ is equivalent to $\phi \cup \perp$, i.e., the only difference is that ϕ should be maintained until τ holds, and thus not indefinitely as was the case with $\Box\phi$.

The fourth temporal goal type is characterised by $\Diamond(\tau \wedge \Box\phi)$. The first condition-action pair ensures that ϕ is maintained indefinitely if the agent's current beliefs entails $\phi \wedge \tau$. However, if either ϕ or τ do not hold, then the temporal goal is suspended and the achieve goal $A(\phi \wedge \tau)$ is added to the basic goal base. This ensures that the agent will pursue the condition before maintaining ϕ indefinitely. The third pair ensures that the goal is activated again once the condition $\phi \wedge \tau$ is satisfied. We adopt a goal to achieve both ϕ and τ , and not just τ , because in the state in which the agent transitions from achieving τ to maintaining ϕ , we want ϕ to be true (so it can be maintained). This is also the reason why the third condition-action pair has a condition of $\phi \wedge \tau$, and not just τ . These points also apply to the following temporal goal types.

The fifth temporal goal type is characterised by $\Diamond(\tau \wedge (\phi \cup \tau'))$. The first condition-action pair indicates that if $\phi \wedge \tau$ holds, then the temporal goal can be dropped and at the same time the basic maintain goal $M(\phi, \tau')$ is adopted. Note that if $\phi \wedge \tau$ holds, the maintain goal $M(\phi, \tau')$ ensures that ϕ is maintained until τ' is achieved. The second condition-action pair covers the situation where either ϕ or τ does not hold. In such a situation, the temporal goal cannot be realized. The temporal goal is therefore suspended, but the basic achievement goal $A(\phi \wedge \tau)$ is added in order to achieve the condition for the first condition-action pair and the goal is activated again once this condition is satisfied.

Finally, the sixth temporal goal type is characterized by $\Box(\tau \rightarrow (\phi \cup \tau'))$. The first condition-action pair specifies that when $\phi \wedge \tau$ is entailed by an agent's belief base the temporal goal can be suspended (not dropped) and the basic maintain goal $M(\phi, \tau')$ adopted. The adopted maintain goal ensures the maintenance of ϕ until τ' . Note that the temporal goal can be pursued again since it was suspended, instead of being dropped. The second condition-action pair is to activate the temporal goal. Note that the condition of this pair is the same as the condition of the first pair, but that the first pair is applicable only to active goals and the second only to suspended goals. However, in order to avoid a loop (suspend, activate, suspend, ...) we impose an additional condition that a suspended goal of this type cannot be activated again until the plan generated for the goal has been performed. Formalising this restriction is straightforward but is omitted for space reasons.

4.2 Life Cycle for Basic Goals

Generic condition-action pairs are also assigned to basic goals when they are adopted for a temporal goal. As we will see later, these condition-action pairs implement the generic relation between beliefs and goals, e.g., an achieve goal $A(\phi)$ is dropped when ϕ is believed and a maintain goal $M(\phi, \tau)$ is dropped if τ is believed.

Note that the second condition-action pair for basic maintenance

BGoal	Condition – Action Pairs λ
$A(\phi)$	$\{\langle\phi, Drop\rangle\}$
$M(\phi, \tau)$	$\{\langle\neg\phi \wedge \neg\tau, Suspend\rangle, \langle\phi \vee \tau, Activate\rangle, \langle\tau, Drop\rangle\}$

Figure 4: Generic Condition-Action Pairs for Basic Goals

goals, which activates the goal, is only applicable to suspended goals (as defined in the transition rules in Section 5.2). On the other hand, the last condition-action pair, which drops a basic maintenance goal, can only be applied to an active basic maintenance goal. It should also be observed that our assumption about correct operationalisation of maintain goals implies that the condition of the suspend action is never satisfied, and therefore the basic maintain goal never gets into the suspended state. This condition-action pair is used in cases where the assumption that maintenance goals are correctly operationalised does not hold.

Additionally, a set of domain related condition-action pairs are assigned to each basic goal. Like temporal goals, domain related condition-action pairs for basic goals should be used with care since otherwise they may cause undesirable behavior. The condition-action pairs for basic goals may seem redundant as they do not contribute to the operationalisation of temporal goals. However, these condition-action pairs generalise the presented framework, thus allowing for the design of arbitrary basic goal behaviors, which may be useful for a variety of domain dependent applications. For example, a robot with an achieve goal to be at a certain position will suspend its achieve goal when its battery charge is not sufficient.

5. OPERATIONAL SEMANTICS

The operationalization of goal types is accomplished by operational semantics, which indicate possible transitions between agent configurations due to goal processing.

Definition 1 *The agent configuration is defined as a tuple $\langle\sigma, \gamma_t, \gamma_b\rangle$, where σ is the agent's belief base (a finite set of propositional atoms), γ_t is the temporal goal base (a finite set of triples of the form $(\phi, state, \Delta)$ where ϕ is a temporal formula, $state$ is the state of the temporal goal (init, active or suspended), and Δ is the set of condition-action pairs governing the life cycle), and γ_b is the basic goal base (a finite set of triples of the form $(g, state, \Delta)$ where g is one of $A(\phi)$ or $M(\phi, \tau)$, or $M(\phi, \perp)$, and the remaining components are the same as for the temporal goal base). $A(\phi)$ and $M(\phi, \tau)$ denote goals to achieve and maintain the non-temporal formula ϕ , respectively. The maintain goal $M(\phi, \tau)$ has an additional argument, a proposition τ , that indicates the deadline until which ϕ should be maintained.*

The operational semantics defines how the agent pursues complex temporal goals in terms of the pursuit of basic achievement and maintenance goals. How the agent pursues basic achievement and maintenance goals is not defined here, and can be found elsewhere (e.g. [19]). We make assumptions about the pursuit of achievement and maintenance goals being operationalised correctly, and then show that, given these assumptions, the semantics given here correctly achieve complex temporal goals.

5.1 Transition Rules for Temporal Goals

Below, we specify transition rules for individual temporal goals, and after that transition rules that lift these to sets of temporal goals. Let ϕ be a temporal goal, δ_d a set of domain related condition-action pair for ϕ , and $\delta_g(\phi)$ be the set of generic condition-action pairs as defined in Section 4.1. Let λ be the generic condition-action pair

for basic goals as defined in Figure 4. We define $+(X, \gamma_b)$ (i.e., γ_b extended with basic goal X) as follows (note that we do not add any domain related condition-action pair to basic goals).

- $+(\emptyset, \gamma_b) = \gamma_b$
- $+(X, \gamma_b) = \gamma_b \cup \{(X, \text{active}, \lambda)\}$

The first two rules below (*Adopt1* and *Adopt2*) define the adoption of a temporal goal $\langle \phi, \text{init}, \delta_d \rangle$, firstly in the case where there is a condition-action pair to suspend the goal while adopting a basic goal, and secondly where there is no applicable condition-action pair to suspend the goal (no basic goal is added). Note that in both cases, temporal goals are adopted entering in the suspended state. Temporal goals are initially suspended in order to ensure that their corresponding maintain goal is added to the basic goal base only in states where their maintain condition is satisfied. This can be verified by observing the condition-action pairs that add maintain goals to the basic goal base in Figure 3. Note that the application of the first transition rule adds a maintain goal to the basic goal base only for the sixth goal type and only when the condition of the to be added maintain goal is satisfied. In other cases, the temporal goals that would add a maintain goal are suspended without adding the maintain goal to the basic goal base until the maintain conditions are satisfied.

In contrast to temporal goals, basic goals are adopted in an active state (see above in the definition of $+(X, \gamma_b)$). This is because achieve goals can always be activated, i.e., there is no reason why they should be suspended at the start. A maintain goal can also start in an active state since its adoption condition ensures its maintain condition holds as explained above.

It is also important to notice that in the first two transition rules we use generic condition-action pairs only from δ_g (and not from Δ). This is because domain related condition-action pairs are only used for activation and dropping of the goals, not for their adoption. Finally, observe that the first five transition rules allow transitions from one single temporal goal to a set of temporal goals. This means that these transition rules cannot be applied consecutively. However, the last transition rule is designed to manage the processing of a set of temporal goals in terms of transitions that are derivable from the first five transition rules.

$$\frac{\langle c, \text{Suspend}(X) \rangle \in \delta_g(\phi) \quad \sigma \models c}{\langle \sigma, (\phi, \text{init}, \delta_d), \gamma_b \rangle \rightarrow \langle \sigma, \{(\phi, \text{susp}, \delta_d \cup \delta_g(\phi))\}, +(X, \gamma_b) \rangle} \text{Adopt1}$$

$$\frac{\neg \exists (c, \text{Suspend}(X)) \in \delta_g(\phi) : \sigma \models c}{\langle \sigma, (\phi, \text{init}, \delta_d), \gamma_b \rangle \rightarrow \langle \sigma, \{(\phi, \text{susp}, \delta_d \cup \delta_g(\phi))\}, \gamma_b \rangle} \text{Adopt2}$$

The following rule activates a suspended temporal goal.

$$\frac{\langle c, \text{Activate} \rangle \in \Delta \quad \sigma \models c}{\langle \sigma, (\phi, \text{susp}, \Delta), \gamma_b \rangle \rightarrow \langle \sigma, \{(\phi, \text{active}, \Delta)\}, \gamma_b \rangle} \text{Activate}$$

The following rule suspends an active temporal goal and (possibly) adds a basic goal to the basic goal base.

$$\frac{\langle c, \text{Suspend}(X) \rangle \in \Delta \quad \sigma \models c}{\langle \sigma, (\phi, \text{active}, \Delta), \gamma_b \rangle \rightarrow \langle \sigma, \{(\phi, \text{susp}, \Delta)\}, +(X, \gamma_b) \rangle} \text{Suspend}$$

The following rule drops a temporal goal and (possibly) adds a basic goal to the basic goal base.

$$\frac{\langle c, \text{Drop}(X) \rangle \in \Delta \quad \sigma \models c}{\langle \sigma, (\phi, \text{active}, \Delta), \gamma_b \rangle \rightarrow \langle \sigma, \{\}, +(X, \gamma_b) \rangle} \text{Drop}$$

The following transition rule specifies how the above rules for single temporal goals (denoted as g) can be lifted to temporal goal

bases. Note that whereas g is a single goal (a tuple), g' is a set of goals (either singleton or empty).

$$\frac{g \in \gamma_t \quad \langle \sigma, g, \gamma_b \rangle \rightarrow \langle \sigma, g', \gamma'_b \rangle}{\langle \sigma, \gamma_t, \gamma_b \rangle \rightarrow \langle \sigma, (\gamma_t \setminus \{g\}) \cup g', \gamma'_b \rangle} \text{Lift}$$

5.2 Transition Rules for Basic Goals

The following rules specify the life cycle of a basic goal X . The *DropBasic* rule indicates that if a basic goal is in an active state, then it can be dropped if there is a corresponding condition-action pair for which the condition is satisfied in the current state and the action is a drop action. It should be noted that for a basic achievement goal we have $\langle \phi, \text{Drop} \rangle$, which indicates that the basic goal $A(\phi)$ can be dropped when ϕ holds. Similarly, for a basic maintain goal we have $\langle \tau, \text{Drop} \rangle$, which indicates that the maintain goal $M(\phi, \tau)$ can be dropped if τ holds in the current state.

$$\frac{\langle c, \text{Drop} \rangle \in \Delta \quad \sigma \models c}{\langle \sigma, \gamma_t, (X, \text{active}, \Delta) \rangle \rightarrow \langle \sigma, \gamma_t, \{\} \rangle} \text{DropBasic}$$

The following transition rule (*SuspB*) specifies that a goal in an active state can be suspended. Note that the corresponding condition-action pair for a maintain goal indicates that a maintain goal can be suspended if neither ϕ nor τ hold in the current state.

$$\frac{\langle c, \text{Suspend} \rangle \in \Delta \quad \sigma \models c}{\langle \sigma, \gamma_t, (X, \text{active}, \Delta) \rangle \rightarrow \langle \sigma, \gamma_t, \{(X, \text{susp}, \Delta)\} \rangle} \text{SuspB}$$

The next transition rule is designed to manage the transition of a basic goal from suspended to active state. Note that the corresponding condition-action pair for a maintain goal states that the basic maintain goal $M(\phi, \tau)$ can be activated if either ϕ or τ hold in the current state.

$$\frac{\langle c, \text{Activate} \rangle \in \Delta \quad \sigma \models c}{\langle \sigma, \gamma_t, (X, \text{susp}, \Delta) \rangle \rightarrow \langle \sigma, \gamma_t, \{(X, \text{active}, \Delta)\} \rangle} \text{ActivB}$$

Finally, as for temporal goals, we have a transition rule that specifies how the above rules for single basic goals (denoted as g) can be lifted to basic goal bases. Note again that whereas g is a single basic goal (a tuple), g' is a set of basic goals (either singleton or empty).

$$\frac{g \in \gamma_b \quad \langle \sigma, \gamma_t, g \rangle \rightarrow \langle \sigma, \gamma_t, g' \rangle}{\langle \sigma, \gamma_t, \gamma_b \rangle \rightarrow \langle \sigma, \gamma_t, (\gamma_b \setminus \{g\}) \cup g' \rangle} \text{LiftB}$$

5.3 Properties

In the following, we use P to denote the set of non-temporal propositional formulae, and we use \models , \models_{cwa} , and \models_{LTL} to respectively denote propositional entailment, propositional entailment based on the closed-world assumption, and LTL entailment.

We define the transition system Σ to include the rules defined earlier in this section. It is important to observe that the transition system Σ contains other transition rules in addition to those presented in sections 5.1 and 5.2. In particular, Σ is assumed to contain transition rules for action execution, which may change the belief states. As the application of transition rules may be interleaved, possible belief changes may influence the goal life cycles. One assumption that we make about the details of transition system Σ , is that the rules defined earlier in this section have a higher priority than other rules. So for instance, if there are two applicable transition rules, say one for deriving/allowing an action execution transition, and the other for the application of a condition-action pair of a goal, then the second transition rule is applied to derive/allow the transition of goal life cycle. This assumption is crucial to show the properties of our transition system in the rest of this section.

Definition 2 Let $S = \langle \sigma^1, \gamma_i^1, \gamma_b^1 \rangle \rightarrow \langle \sigma^2, \gamma_i^2, \gamma_b^2 \rangle \rightarrow \dots$ be an infinite sequence of configurations generated by the transition system Σ . In this sequence, the transition $\langle \sigma^i, \gamma_i^i, \gamma_b^i \rangle \rightarrow \langle \sigma^{i+1}, \gamma_i^{i+1}, \gamma_b^{i+1} \rangle$ is derived by the application of a transition rule of Σ . The sequence S is called a trace of Σ . The trace S of Σ is called a fair trace if it is generated by a fair run of the transition system, that is, a run in which every transition rule that is enabled infinitely often, is also applied infinitely often.

Note that every finite trace is a fair trace. Furthermore, the priority assumption does not affect fairness, since in any given configuration, there is only a finite number of goal-related transitions that can be applied. In the following, we consider only infinite traces (which can be guaranteed by adding an ‘‘idling’’ rule that transitions a configuration to itself if no other transition rules are applicable). We use σ^i , γ_b^i , and γ_i^i to denote the ingredients of the i th state of a trace S .

Definition 3 Let S be a trace of Σ . We define $B(S)$, called the belief trace of S , to be the LTL-trace $s = s^1 s^2 \dots$ (where s^i is a state assigning a truth value to each atomic proposition), if the following condition holds:

$$\forall \phi \in P, \forall i \in \mathbb{N} : \sigma^i \models_{cwa} \phi \Leftrightarrow s^i \models \phi$$

Furthermore, observe that since ϕ is non-temporal, we also have that $s^i \models \phi \Leftrightarrow B(S), i \models_{LTL} \phi$.

The following proposition states that for every trace there is one unique belief trace possible.

Proposition 1 Let S be a trace of Σ . The belief trace $B(S)$ is uniquely determined by S .

Proof: This proposition is the direct consequence of matching belief bases σ^i with states s^i using the closed-world assumption. I.e., state s^i assigns the truth value of propositions based on their truth values in σ^i using closed-world assumption.

Assumption 1 The achieve goal $A(\phi)$ is properly operationalized by a transition system S if the following condition holds for every trace S of Σ :

$$\forall i : (A(\phi), \text{active}, \Delta) \in \gamma_b^i \Rightarrow \exists j \geq i : \sigma^j \models_{cwa} \phi$$

Assumption 2 The maintain goal $M(\phi, \tau)$ is properly operationalized by a transition system S if the following condition holds for every trace S of Σ :

$$\begin{aligned} \forall i : (M(\phi, \tau), \text{active}, \Delta) \in \gamma_b^i \Rightarrow \\ (\exists j \geq i : \sigma^j \models_{cwa} \tau) \wedge (\forall i \leq k < j : \sigma^k \models_{cwa} \phi) \\ \text{or } \tau = \perp \wedge (\forall k \geq i : \sigma^k \models_{cwa} \phi) \end{aligned}$$

The following propositions show that the operational semantics defined in section 5 correctly operationalize complex temporal goals (in γ_i) using the basic achievement and maintenance goals (in γ_b). Generally, correct realisation is the property that if a temporal LTL formula χ is in the temporal goal base of state i and is active, formally $(\chi, \text{active}, \Delta) \in \gamma_i^i$, then $B(S), i \models_{LTL} \chi$. However, since this only holds for the particular goal patterns that have been operationalized, we prove this for each case separately. All propositions are based on assumptions 1 and 2.

Proposition 2 If $(\Diamond \phi, \text{active}, \Delta) \in \gamma_i^i$, then $B(S), i \models_{LTL} \Diamond \phi$ for all fair traces S of Σ .

Proof: Case 1: Assume that $\sigma^i \not\models_{cwa} \phi$. By the definition of the condition-action pairs for $\Diamond \phi$ and the transition rule for Drop, we

have that, since $\Diamond \phi \in \gamma_i^i$, we must have that $A(\phi) \in \gamma_b^{i+1}$. Therefore by assumption 1 (and definition 3) we have that $\exists j \geq i + 1 : B(S), j \models_{LTL} \phi$ and hence by the semantics of LTL that $B(S), i \models_{LTL} \Diamond \phi$. Case 2: Assume that $\sigma^i \models_{cwa} \phi$. Hence $B(S), i \models_{LTL} \phi$ and trivially $B(S), i \models_{LTL} \Diamond \phi$.

The following proposition states that a property can be maintained if it already holds.

Proposition 3 If $(\Box \phi, \text{active}, \Delta) \in \gamma_i^i$ and $\sigma^i \models_{cwa} \phi$, then $B(S), i \models_{LTL} \Box \phi$ for all fair traces S of Σ .

Proof: Since $\Box \phi \in \gamma_i^i$, by the definition of the condition-action rules and the transition rule for Drop, we have that $M(\phi, \perp) \in \gamma_b^{i+1}$. By assumption 2 (and definition 3), since $\sigma^i \models_{cwa} \phi$ and $\forall j \geq i + 1 : \sigma^j \models_{cwa} \phi$, we have $\forall j \geq i : B(S), j \models_{LTL} \phi$ and hence $B(S), i \models_{LTL} \Box \phi$.

The following proposition shows that $\phi \cup \tau$ is correctly operationalized. It assumes that $\tau \neq \perp$, since $\phi \cup \perp$ is false in LTL.

Proposition 4 If $(\phi \cup \tau, \text{active}, \Delta) \in \gamma_i^i$ (where $\tau \neq \perp$), and $\sigma^i \models_{cwa} \phi$, then $B(S), i \models_{LTL} \phi \cup \tau$ for all fair traces S of Σ .

Proof (sketch): Since $(\phi \cup \tau) \in \gamma_i^i$, by the definition of the condition-action pairs and of the transition rules, we have $M(\phi, \tau) \in \gamma_b^{i+1}$. By assumption 2 the trace S at configuration $i + 1$ satisfies $\exists j \geq i + 1 : \sigma^j \models_{cwa} \tau \wedge \forall k : i + 1 \leq k < j : \sigma^k \models_{cwa} \phi$. From this condition together with the definition 3 and the fact that $B(S), i \models_{LTL} \phi$, it is easy to see that $B(S), i \models_{LTL} \phi \cup \tau$.

Proposition 5 If $(\Diamond(\tau \wedge \Box \phi), \text{active}, \Delta) \in \gamma_i^i$, then $B(S), i \models_{LTL} \Diamond(\tau \wedge \Box \phi)$ for all fair traces S of Σ .

Proof (sketch): We consider two cases in configuration i . Case 1: $\sigma^i \not\models_{cwa} \phi \wedge \tau$ (either ϕ or τ is not believed in the current configuration). Since $\Diamond(\tau \wedge \Box \phi) \in \gamma_i^i$, by the definition of the condition-action pairs and of the transition rules, we have $A(\phi \wedge \tau) \in \gamma_b^{i+1}$. By assumption 1 (and definition 3), we have that $\exists j \geq i + 1 : B(S), j \models_{LTL} \phi \wedge \tau$. Since $\Diamond(\tau \wedge \Box \phi)$ is never removed from the temporal goal base (the condition-action pairs always suspend, and never drop it), we have $\Diamond(\tau \wedge \Box \phi) \in \gamma_i^i$. By the definition of the condition-action pairs and of the transition rules, and because $\sigma^j \models_{cwa} \phi \wedge \tau$, we have $M(\phi, \perp) \in \gamma_b^{j+1}$. From, $\sigma^j \models_{cwa} \phi$, assumption 2 and the definition 3, we have $B(S), j \models_{LTL} \Box \phi$ and hence $B(S), i \models_{LTL} \Diamond(\tau \wedge \Box \phi)$. Case 2: $\sigma^i \models_{cwa} \phi \wedge \tau$. It is easy to see that the proposition holds in this case by having $i = j$ in the proof sketch of case 1.

Proposition 6 If $(\Diamond(\tau \wedge (\phi \cup \tau')), \text{active}, \Delta) \in \gamma_i^i$, then $B(S), i \models_{LTL} \Diamond(\tau \wedge (\phi \cup \tau'))$ for all fair traces S of Σ .

Proof (sketch): The proof is similar to the proof of the previous proposition.

The following proposition assumes that $\tau \rightarrow \phi$. The justification for this assumption is that we want to show that the temporal goal is correctly realised. In the case where τ becomes true but ϕ does not hold, the temporal goal immediately fails, i.e. no operationalisation is able to realise the goal. We thus exclude this case.

Proposition 7 If $(\Box(\tau \rightarrow (\phi \cup \tau')), \text{active}, \Delta) \in \gamma_i^i$ and $\tau \rightarrow \phi$, then $B(S), i \models_{LTL} \Box(\tau \rightarrow (\phi \cup \tau'))$ for all fair traces S of Σ .

Proof (sketch): We want to show that $B(S), i \models_{LTL} \Box(\tau \rightarrow (\phi \cup \tau'))$, i.e. that for any $j \geq i$, we have $B(S), j \models_{LTL} (\tau \rightarrow (\phi \cup \tau'))$. There are two cases. Case 1: $\sigma^j \not\models_{cwa} \tau$. In this case the implication trivially holds. Case 2: $\sigma^j \models_{cwa} \tau$. Since $\tau \rightarrow \phi$, we have $\sigma^j \models_{cwa} \phi$. Since the temporal goal is always suspended and never dropped, $\Box(\tau \rightarrow (\phi \cup \tau')) \in \gamma_i^i$ for all $j \geq i$. Thus, by the definition of

the condition-action pairs and of the transition rules, and since $\sigma^j \models_{cwa} \tau \wedge \phi$, we have $M(\phi, \tau') \in \gamma_b^{j+1}$. By assumption 2 the trace S at configuration $j+1$ ensures that ϕ will be entailed by the belief base until τ' is entailed. Because $\sigma^j \models_{cwa} \phi$, trace S at configuration j ensures that ϕ is entailed by the belief base until τ' is entailed, i.e. $B(S, j \models_{LTL} \phi \cup \tau'$, from which $B(S, j \models_{LTL} (\tau \rightarrow (\phi \cup \tau'))$ trivially follows.

6. DISCUSSION

In this paper we built on a temporal logic view of agent goals by defining new, novel, goal types, and showing how these new goal types could be operationalized in terms of existing base goal types (achievement and maintenance). The operationalization is based on a goal life cycle where transitions between states of goals are governed by condition-action pairs. We show that the operationalization realizes traces on which the temporal goals are satisfied, assuming satisfaction of the basic goal types. Through this we have provided a flexible framework for operationalizing rich goal types. Other goal types can be added by providing their translation to the basic goal types. Although we have provided several examples in the paper, future work will have to show which goal types are particularly useful in practice. This may also depend on the domain that is modelled.

An important topic for future work to allow gaining more practical experience with the framework is implementing it on top of conventional agent-oriented programming languages, such as 2APL [6], Jadex [15], Jason [3], JACK [5] etc. It is interesting to note that the proposed framework appears to be a good match with rule-based systems, as used in Opal [20]. Also, we aim to extend the framework to include subgoals along the lines of [14], and then using this as a basis for incorporating goal suspension/resumption and abortion (based on [17, 18]).

7. REFERENCES

- [1] F. Bacchus and F. Kabanza. Planning for temporally extended goals. In *Proceedings of the 13th National Conference on Artificial Intelligence (AAAI'96)*, pages 1215–1222, 1996.
- [2] C. Baral and J. Zhao. Non-monotonic temporal logics for goal specification. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 236–242, 2007.
- [3] R. H. Bordini, J. F. Hübner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*. Wiley, 2007. ISBN 0470029005.
- [4] L. Braubach and A. Pokahr. Representing long-term and interest BDI goals. In *Programming Multi-Agent Systems (ProMAS)*, 2009.
- [5] P. Busetta, R. Rönnquist, A. Hodgson, and A. Lucas. JACK Intelligent Agents - Components for Intelligent Agents in Java. Technical report, Agent Oriented Software Pty. Ltd, Melbourne, Australia, 1998. Available from <http://www.agent-software.com>.
- [6] M. Dastani. 2APL: a practical agent programming language. *Autonomous Agents and Multi-Agent Systems*, 16(3):214–248, 2008.
- [7] M. Dastani, M. B. van Riemsdijk, and J.-J. Ch. Meyer. Goal types in agent programming. In *Proceedings of the 17th European Conference on Artificial Intelligence 2006 (ECAI'06)*, volume 141 of *Frontiers in Artificial Intelligence and Applications*, pages 220–224. IOS Press, 2006.
- [8] S. Duff, J. Harland, and J. Thangarajah. On proactivity and maintenance goals. In *Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 1033–1040, Hakodate, 2006.
- [9] E. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 996–1072. Elsevier, Amsterdam, 1990.
- [10] M. Fisher and A. Hepple. Executing logical agent specifications. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*, volume 2, chapter 1, pages 3–29. Springer, 2009.
- [11] K. Hindriks and M. B. van Riemsdijk. Satisfying maintenance goals. In *Declarative Agent Languages and Technologies (DALT'07)*, volume 4897 of *LNAI*, pages 86–103. Springer, 2008.
- [12] K. V. Hindriks, W. van der Hoek, and M. B. van Riemsdijk. Agent programming with temporally extended goals. In *Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 137–144. IFAAMAS, 2009.
- [13] S. M. Khan and Y. Lespérance. A logical account of prioritized goals and their dynamics. In G. Lakemeyer, L. Morgenstern, and M. A. Williams, editors, *Proc. of the 9th International Symposium on Logical Formalizations of Commonsense Reasoning*, pages 85–90, 2009.
- [14] M. Morandini, L. Penserini, and A. Perini. Operational semantics of goal models in adaptive agents. In C. Sierra, C. Castelfranchi, K. S. Decker, and J. S. Sichman, editors, *Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 129–136. IFAAMAS, 2009.
- [15] A. Pokahr, L. Braubach, and W. Lamersdorf. Jadex: a BDI reasoning engine. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah Seghrouchni, editors, *Multi-Agent Programming: Languages, Platforms and Applications*. Springer, Berlin, 2005.
- [16] S. Shapiro and G. Brewka. Dynamic interactions between goals and beliefs. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2625–2630, 2007.
- [17] J. Thangarajah, J. Harland, D. Morley, and N. Yorke-Smith. Aborting goals and plans in BDI agents. In *Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2007.
- [18] J. Thangarajah, J. Harland, D. Morley, and N. Yorke-Smith. Suspending and resuming tasks in intelligent agents. In Padgham, Parkes, Müller, and Parsons, editors, *Autonomous Agents and Multi-Agent Systems (AAMAS)*, 2008.
- [19] M. B. van Riemsdijk, M. Dastani, and M. Winikoff. Goals in agent systems: A unifying framework. In Padgham, Parkes, Müller, and Parsons, editors, *Autonomous Agents and Multi-Agent Systems (AAMAS)*, pages 713–720. IFAAMAS, 2008.
- [20] M. Wang, M. Nowostawski, and M. K. Purvis. Declarative agent programming support for a FIPA-compliant agent platform. In R. H. Bordini, M. Dastani, J. Dix, and A. El Fallah-Seghrouchni, editors, *ProMAS*, volume 3862 of *LNCS*, pages 252–266. Springer, 2005.
- [21] M. Winikoff, L. Padgham, J. Harland, and J. Thangarajah. Declarative & procedural goals in intelligent agent systems. In *Proceedings of the Eighth International Conference on Principles of Knowledge Representation and Reasoning (KR)*, Toulouse, France, Apr. 2002.

Bounded Rationality

Expert-Mediated Search

Meenal Chhabra
Rensselaer Polytechnic Inst.
Dept. of Computer Science
Troy, NY, USA
chhabm@cs.rpi.edu

Sanmay Das
Rensselaer Polytechnic Inst.
Dept. of Computer Science
Troy, NY, USA
sanmay@cs.rpi.edu

David Sarne
Bar-Ilan University
Dept. of Computer Science
Ramat Gan, Israel
sarned@cs.biu.edu

ABSTRACT

Increasingly in both traditional, and especially Internet-based marketplaces, knowledge is becoming a traded commodity. This paper considers the impact of the presence of knowledge-brokers, or experts, on search-based markets with noisy signals. For example, consider a consumer looking for a used car on a large Internet marketplace. She sees noisy signals of the true value of any car she looks at the advertisement for, and can disambiguate this signal by paying for the services of an expert (for example, getting a Carfax report, or taking the car to a mechanic for an inspection). Both the consumer and the expert are rational, self-interested agents. We present a model for such search environments, and analyze several aspects of the model, making three main contributions: (1) We derive the consumer's optimal search strategy in environments with noisy signals, with and without the option of consulting an expert; (2) We find the optimal strategy for maximizing the expert's profit; (3) We study the option of market designers to subsidize search in a way that improves overall social welfare. We illustrate our results in the context of a plausible distribution of signals and values.

Categories and Subject Descriptors

J.4 [Social and Behavioral Sciences]: Economics

General Terms

Algorithms, Economics

Keywords

Economically-motivated agents, Modeling the dynamics of MAS

1. INTRODUCTION

In many multi-agent system (MAS) settings, agents engage in *one-sided search* [13, 6]. This is a process in which an agent faces a stream of opportunities that arise sequentially, and the process terminates when the agent picks one of those opportunities. A classic example is a consumer looking to buy a used car. She will typically investigate cars one at a time until deciding upon one she wants. Similar settings

can be found in job-search, house search and other applications [12, 15]. In modern electronic marketplaces, search is likely to become increasingly important as, with the proliferation of possible sellers of a good, consumers will turn to artificial agents, whom we term “searchers,” to find the best prices or values for items they are interested in acquiring.

The decision making complexity in one-sided search usually arises from the fact that there is a cost incurred in finding out the true value of any opportunity encountered. For example, there is a cost to arranging a meeting to test drive a car you are considering purchasing. The searcher thus needs to trade off the potential benefit of continuing to search and seeing a more valuable opportunity with the costs incurred in doing so. The optimal stopping rule for such search problems has been widely studied, and is often a *reservation strategy*, where the searcher should terminate search upon encountering an opportunity which has a value above a certain reservation value or threshold [18, 13]. Most models assume that the searcher obtains the exact true value of the opportunities it encounters. However, in many realistic settings, search is inherently noisy and searchers may only obtain a noisy signal of the true value. For example, the drivetrain of a used car may not be in good condition, even if the body of the car looks terrific. The relaxation of the assumption of perfect values not only changes the optimal strategy for a searcher, it also leads to a niche in the marketplace for new knowledge-brokers. The knowledge brokers, or *experts*, are service providers whose main role is to inform consumers or searchers about the values of opportunities.

An expert offers the searcher the option to obtain a more precise estimate of the value of an opportunity in question, in exchange for the payment of a fee (which covers the cost of providing the service as well as the profit of the expert). To continue with the used car example, when the agent is intrigued by a particular car and wants to learn more about it, she could take the car to a mechanic who could investigate the car in more detail to make sure it is not a lemon. The expert need not be a mechanic – it could be, for example, an independent agency, like Carfax, that monitors the recorded history of transactions, repairs, claims, etc. on cars. It can also be a repeated visit of the searcher to see the car, possibly bringing an experienced friend for a more thorough examination. In all these cases, more accurate information is obtained for an additional cost (either monetary or equivalent).

In this paper we investigate optimal search and mechanism design in environments where searchers observe noisy signals and can obtain (i.e., query the expert for) the ac-

Cite as: Expert-Mediated Search, Meenal Chhabra, Sanmay Das, and David Sarne, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 415-422. Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

tual values for a fee. Our main contributions are threefold: First, we introduce a specific model of one-sided search with noisy signals and prove that the optimal search rule, for a large class of real-life settings, is reservation-value based. Second, we formally introduce the option of consulting an expert in such noisy search environments, and derive the optimal strategy for the searcher given a cost of consulting the expert, as well as the profit-maximizing price for the expert to charge for its services. As part of the analysis we prove that under the standard assumption that higher signals are “good news” (i.e., the distribution of the true value conditional on a higher signal stochastically dominates the distribution conditional on a lower signal), the optimal search strategy is characterized by a “double reservation value” strategy, wherein the searcher rejects all signals below a certain threshold, resuming search, and accepts all signals above another threshold, thus terminating search, without querying the expert; the agent queries the expert for all signals that are between the two thresholds. Finally, we study a market design mechanism (introduction of a subsidy) with the potential to improve social welfare in such domains. These mechanisms may be implemented by either an electronic marketplace that employs the expert (for example a website for used cars that has a relationship with a provider of car history reports), or an entity with regulatory power like the government. Along with the general theory, we illustrate our results in a specific, plausible distributional model, in which the true value is always bounded by the signal value, and the probability monotonically decays as the discrepancy between the two increases.

2. THE GENERAL MODEL

The standard one-sided search problem [13] considers an agent or *searcher* facing an infinite stream of opportunities from which she needs to choose one. While the specific value v of each future opportunity is unknown to the searcher, she is acquainted with the (stationary) probability distribution function from which opportunities are drawn, denoted $f_v(x)$. The searcher can learn the value of an opportunity for a cost c_s (either monetary or in terms of resources that need to be consumed for this purpose) and her goal is to maximize the net benefit, defined as the value of the opportunity eventually picked minus the overall cost incurred during the search. Having no *a priori* information about any specific opportunity, the searcher reviews the opportunities she encounters sequentially and sets her optimal stopping rule. The stopping rule specifies when to terminate and when to resume search, based on the opportunities encountered.

Our model relaxes the standard assumption that the searcher receives the exact true value of an opportunity. Instead, we assume that the searcher receives, at cost c_s , a noisy signal s , correlated with the true value according to a known probability density function $f_s(s|v)$. In addition, the searcher may query and obtain from a third party (the *expert*) the true value v of an opportunity for which signal s was received, by paying an additional fee c_e . The goal of the searcher is to maximize the total utility received i.e., the expected value of the opportunity eventually picked minus the expected cost of search and expert fees paid along the way.

The first question that arises is how to characterize the optimal strategy for the searcher. A second question is how the expert sets her service fee c_e . In this paper we consider a monopolist provider of expert services. The searcher’s opti-

mal strategy is directly influenced by c_e , and thus implicitly determines the expected number of times the services of the expert are required, and thus the expert’s revenue. The problem can be thought of as a Stackelberg game [5] where the expert is the first mover, and wants to maximize her profits with respect to the fee c_e she charges searchers.

The new search model raises interesting new questions about market design. Assuming exogeneity of opportunities, social welfare, denoted W , is a function of the expected value to searchers and the expected profit of the expert. (We abstract away from modeling the existence of “sellers” of opportunities, instead viewing them as exogenous, or else as being offered at some “fair price” by the seller.) We assume that the provider of expert services has already performed the “startup work” necessary, and only pays a marginal cost d_e per query, and that social welfare is additive. Since the process scales up linearly in the number of searchers, we can simply consider the interactions involving a single searcher and the expert. The social welfare is then the sum of the expected net benefit to the searcher and the expected profit of the expert. It turns out that social welfare can be significantly affected (and improved) if the market designer (or a regulator like the government) subsidizes queries by compensating the expert in order to reduce query costs to the searcher.

We now turn to developing the mathematical machinery to address these problems.

3. OPTIMAL POLICIES

In this section, we analyze the searcher’s optimal search strategy and her expected use of the expert’s services, given the fee c_e set by the expert. The analysis builds on the trivial non-noisy model and gradually adds the complexities of signals and having the expert option. From the searcher’s optimal search strategy we derive the expert’s expected benefits as a function of the fee she sets, enabling maximization of the expert’s revenue.

One-Sided Search.

The optimal search strategy for the standard model, where the actual value of an opportunity can be obtained at cost c_s , can be found in the extensive literature of search theory [13, 6]. In this case, the searcher follows a reservation-value rule: she reviews opportunities sequentially (in random order) and terminates the search once a value greater than a reservation value x^* is revealed, where the reservation value x^* satisfies:

$$c_s = \int_{y=x^*}^{\infty} (y - x^*) f_v(y) dy \quad (1)$$

Intuitively, x^* is the value where the searcher is precisely indifferent: the expected marginal benefit from continuing search and obtaining the value of the next opportunity exactly equals the cost of obtaining that additional value. The reservation property of the optimal strategy derives from the stationarity of the problem — resuming the search places the searcher at the same position as at the beginning of the search [13]. Consequently, a searcher that follows a reservation value strategy will never decide to accept an opportunity she has once rejected and the optimal search strategy is the same whether or not recall is permitted. The expected number of search iterations is simply the inverse of the success probability, $\frac{1}{1 - F_v(x^*)}$, since this becomes a Bernoulli

sampling process, as opportunities arise independently at each iteration.

One-Sided Search with Noisy Signals.

Before beginning the analysis of search with noisy signals, we emphasize that, given $f_v(x)$ and $f_s(s|v)$, we can also derive the distribution of the signal received from a random opportunity, $f_s(x)$, and the distribution of true values conditional on signals, $f_v(v|s)$ (the conditionals are interchangeable by Bayes' law). In many domains, it may be easier to assess/learn $f_s(s)$ than $f_v(v|s)$ as most past experience involves signals, with the actual value revealed for only a subset of these signals.

When the searcher receives a noisy signal rather than the actual value of an opportunity, there is no guarantee that the optimal strategy is reservation-value based as in the case where values obtained are certain. Indeed, the stationarity of the problem still holds, and an opportunity that has been rejected will never be recalled. Yet, in the absence of any restriction over $f_s(s|v)$, the optimal strategy is based on a set S of signal-value intervals for which the searcher terminates the search. The expected value in this case, denoted $V(S)$, is given by:

$$\begin{aligned} V(S) &= -c_s + \Pr(s \notin S)V(S) + \Pr(s \in S)E[v|s \in S] \\ &= -c_s + V(S) \int_{s \notin S} f_s(s) ds + \int_{s \in S} f_s(s)E[v|s] ds \end{aligned} \quad (2)$$

The fact that the optimal strategy may not be reservation-value based in this case is because there may be no correlation between the signal and the true value of the opportunity. Nevertheless, in most real-life cases, there is a natural correlation between signals and true values. In particular, a fairly weak and commonly used restriction on the conditional distribution of the true value given the signal goes a long way towards allowing us to recapture a simple space of optimal strategies. This is the restriction that higher signal values are "good news" in the sense that when $s_1 > s_2$, the conditional distribution of v given s_1 first-order stochastically dominates that of v given s_2 [19, 14]. The condition requires that given two signals s_1 and s_2 where $s_1 > s_2$, the probability that the actual value is greater than any particular value v is greater for the case where the searcher receives signal s_1 . Formally:

DEFINITION 1. Higher signals are good news (HSGN) assumption: If $s_1 > s_2$, then, $\forall y, F_v(y|s_1) \leq F_v(y|s_2)$.

This enables us to prove the following theorem.

THEOREM 1. For any probability density function $f_v(v|s)$ satisfying the HSGN assumption, the optimal search strategy is a reservation-value rule, where the reservation value, t^* , satisfies:

$$c_s = \int_{s=t^*}^{\infty} (E[v|s] - E[v|t^*]) ds \quad (3)$$

Proof: The proof is based on showing that, if according to the optimal search strategy the searcher should resume her search given a signal s , then she must necessarily also do so given any other signal $s' < s$. Let V denote the expected benefit to the searcher if resuming the search. Since the optimal strategy given signal s is to resume search, we know $V > E[v|s]$. Given the HSGN assumption, $\int_y y f_v(y|s') dy < \int_y y f_v(y|s) dy$ holds for $s' < s$. Therefore, $V > E[v|s']$,

proving that the optimal strategy is reservation-value. Then, the expected value of the searcher when using reservation signal t is given by:

$$\begin{aligned} V(t) &= -c_s + V(t) \int_{s=-\infty}^t f_s(s) ds + \int_{s=t}^{\infty} E[v|s] f_s(s) ds \\ &= \frac{-c_s + \int_{s=t}^{\infty} E[v|s] f_s(s) ds}{1 - F_s(t)} \end{aligned} \quad (4)$$

where $F_s(s)$ is the cumulative distribution function of the signal s . Setting the first derivative according to t of Equation 4 to zero we obtain: $V(t^*) = E[v|t^*]$. The second derivative for t^* that satisfies the latter equality confirms that this is indeed a global maximum. Finally, using integration by parts over the derivative according to t of Equation 4 we obtain Equation 3, and the value t^* can be calculated accordingly. \square

The social welfare W is the expected gain to the searcher from following the optimal strategy, $V(t^*) = E[v|t^*]$. The expected number of search iterations is $\frac{1}{1 - F_s(t^*)}$, since this is a Bernoulli sampling process.

The Expert Option.

The introduction of an expert extends the number of decision alternatives available to the searcher. When receiving a noisy signal of the true value, she can choose to (1) reject the offer without querying the expert, paying search cost c_s to reveal the signal for the next offer; (2) query the expert to obtain the true value, paying a cost c_e , and then make a decision; or (3) accept the offer without querying the expert, receiving the (unknown) true value of the offer. In case (2), there is an additional decision to be made, whether to resume search or not, after the true value v is revealed.

As in the no-expert case, a solution for a general density function $f_v(v|s)$ dictates an optimal strategy of a complex structure. In our case, the optimal strategy will have the form of (S', S'', V) , where: (a) S' is a set of signal intervals for which the searcher should resume her search without querying the expert; (b) S'' is a set of signal intervals for which the searcher should terminate her search without querying the expert (and pick the opportunity associated with this signal); and (c) for any signal that is not in S' or S'' the searcher should query the expert, and terminate the search if the value obtained is above a threshold V , and resume otherwise. The value V is the expected benefit from resuming the search and is given by the following modification of Equation 2:

$$\begin{aligned} V &= -c_s + V \int_{s \in S'} f_s(s) ds - c_e \int_{s \notin \{S', S''\}} f_s(s) ds + \\ &\quad \int_{s \notin \{S', S''\}} f_s(s) \left(V \int_{-\infty}^V f_v(x|s) dx + \int_{x=V}^{\infty} x f_v(x|s) dx \right) ds + \int_{s \in S''} f_s(s) E[v|s] ds \end{aligned} \quad (5)$$

The first element on the right hand side of the equation applies to the case of resuming search, in which case the searcher continues with an expected benefit V . The second element is the expected payment to the expert. The next elements relate to the case where the search is resumed based on the value received from the expert (in which case the expected revenue is once again V) and where the search is

terminated (with the value $E[v|s]$ obtained as the revenue), respectively. Finally, the last element applies to the case where the searcher terminates the search without querying the expert.

We can show that under the HSGN assumption, each of the sets S' and S'' actually contains a single interval of signals, as illustrated in Figure 1.

THEOREM 2. For $f_v(y|s)$ satisfying the HSGN assumption (Definition 1), the optimal search strategy can be described by the tuple (t_l, t_u, V) , where: (a) t_l is a signal threshold below which the search should be resumed; (b) t_u is a signal threshold above which the search should be terminated and the current opportunity picked; and (c) the expert should be queried given any signal $t_l < s < t_u$ and the opportunity should be accepted (and search terminated) if the value obtained from the expert is above the expected value of resuming the search, V , otherwise search should resume (see Figure 1). The values t_l , t_u and V can be calculated from solving the set of Equations 6-8:

$$V = -c_s + V \int_{s=-\infty}^{t_l} f_s(s) ds - c_e \int_{s=t_l}^{t_u} f_s(s) ds + \int_{s=t_l}^{t_u} f_s(s) \left(V \int_{x=-\infty}^V f_v(x|s) dx ds + \int_{x=V}^{\infty} x f_v(x|s) dx ds \right) + \int_{s=t_u}^{\infty} f_s(s) E[v|s] ds \quad (6)$$

$$c_e = \int_{y=V}^{\infty} (y - V) f_v(y|t_l) dy \quad (7)$$

$$c_e = \int_{-\infty}^V (V - y) f_v(y|t_u) dy \quad (8)$$

Proof: The proof extends the methodology used for proving Theorem 1. We first show that if, according to the optimal search strategy the searcher should resume her search given a signal s , then she must also do so given any other signal $s' < s$. Then, we show that if, according to the optimal search strategy the searcher should terminate her search given a signal s , then she must also necessarily do so given any other signal $s'' > s$. Again, we use V to denote the expected benefit to the searcher if resuming the search.

If the optimal strategy given signal s is to resume search then the following two inequalities should hold, describing the superiority of resuming search over terminating search (Equation 9) and querying the expert (Equation 10):

$$V > E[v|s] \quad (9)$$

$$V > -c_e + \int_{y=V}^{\infty} y f_v(y|s) dy + V \int_{y=-\infty}^V f_v(y|s) dy \quad (10)$$

Given the HSGN assumption and since $s' < s$, Equation 9 holds also for s' , and so does Equation 10 (which can be formalized after some mathematical manipulation as: $V > -c_e + V + \int_{y=V}^{\infty} (y - V) f_v(y|s) dy$). The proof for $s'' > s$ is similar: the expected cost of accepting the current opportunity can be shown to dominate both resuming the search and querying the expert. We omit the details because of space considerations. The optimal strategy can thus be described by the tuple (t_l, t_u, V) as stated in the Theorem. Therefore, Equation 5 transforms into Equation 6. Taking the derivative of Equation 6 w.r.t. t_l and equating to zero,

we obtain a unique t_l which maximizes the expected benefit (verified by second derivative), and similarly for t_u . Finally, using integration by parts over the derivatives of Equation 6 w.r.t. t_l and t_u we obtain Equations 7-8. \square

Intuitively, t_l is the point at which a searcher is indifferent between either resuming the search or querying the expert and t_u is the point at which a searcher is indifferent between either terminating the search or querying the expert. The cost of purchasing the expert's services must equal two different things: (1) the expected savings from resuming the search when the actual utility from the current opportunity (which is not known) turns out to be greater than what can be gained from resuming the search (once it is revealed) (this is the condition for t_l); (2) the expected savings from terminating the search in those cases where the actual utility from the current opportunity (once revealed) is less than what can be gained from resuming the search (for t_u).

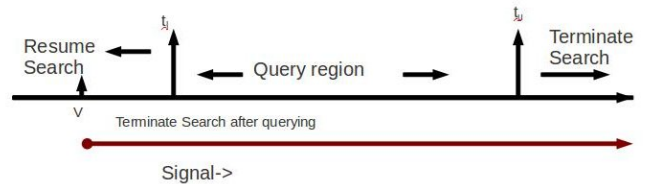


Figure 1: Characterization of the optimal strategy for noisy search with an expert. The searcher queries the expert if $s \in [t_l, t_u]$ and accepts the offer if the worth is greater than the value of resuming the search V . The searcher rejects and resumes search if $s < t_l$ and accepts and terminates search if $s > t_u$, both without querying the expert.

It is notable that there is also a reasonable degenerate case where $t_l = t_u (= t)$. This happens when the cost of querying is so high that it never makes sense to engage the expert's services. In this case, a direct indifference constraint exists at the threshold t , where accepting the offer yields the same expected value as continuing search, so $V = E[v|t]$. This can be solved in combination with Equation 4, since there are now only two relevant variables.

Expected number of queries: The search strategy (t_l, t_u, V) defines how many times the expert's services are consulted. In order to compute the expected number of queries, we consider four different types of transitions in the system. Let A be the probability that the searcher queries the expert and then does not accept, resuming search, B be the probability that the searcher resumes search without querying, C be the probability that the searcher terminates without querying, and D be the probability that the searcher queries the expert and terminates search. Then:

$$A = \Pr(t_l \leq s \leq t_u \text{ and } v < V) \quad (11)$$

$$B = \Pr(s < t_l) \quad (12)$$

$$C = \Pr(s > t_u) \quad (13)$$

$$D = \Pr(t_l \leq s \leq t_u \text{ and } v \geq V) \quad (14)$$

Let P_j denote the probability that the searcher queries the expert exactly j times before terminating. The searcher can terminate search after exactly j queries in one of two ways: either she makes $j - 1$ queries, then queries the expert and chooses to terminate, or she makes j queries and

then chooses to terminate without querying the expert. Factoring in all the possible ways of interleaving j queries with an arbitrary number of times that the searcher chooses to continue search without querying the expert, we get:

$$P_j = \sum_{m=0}^{\infty} \frac{(m+j-1)!}{m!(j-1)!} A^{j-1} B^m D + \sum_{m=0}^{\infty} \frac{(m+j)!}{m!j!} A^j B^m C$$

$$= \frac{A^j}{(1-B)^j} \left(\frac{D}{A} + \frac{C}{1-B} \right)$$

Then the expected number of queries, when charging an expert fee c_e , is given by $\sum_{j=0}^{\infty} j P_j$, yielding

$$\eta_{c_e} = \mathbb{E}(\text{Number of queries} | c_e) = \frac{(1-B)D + CA}{(1-B-A)^2} \quad (15)$$

Expected number of opportunities examined: Using the same notation as above, we see that the probability of terminating the search at any iteration is $C + D$, and these are independent Bernoulli draws at each opportunity. Therefore the expectation of the number of opportunities examined is simply $\eta_s = 1/(C + D)$.

Expected profit of the expert: Let d_e denote the marginal cost of the service the expert is providing. The expected profit of the expert is then simply

$$\pi_e = \mathbb{E}(\text{Profit}) = (c_e - d_e)\eta_{c_e}$$

The expert can maximize the above expression with respect to c_e (η_{c_e} decreases as c_e increases) to find the profit maximizing price to charge searchers.

Social Welfare: The social welfare is given by the sum of all parties involved, thus far just the searcher and the expert. Of course this generalizes to multiple agents as well, since each search process would be independent. We define:

$$W = V_{c_e^*} + \pi_e \quad (16)$$

where c_e^* is the fee that maximizes the expert's profit.

4. MARKET DESIGN

Above, we have described the basics of search in such expert-mediated markets. In this section, we describe possible uses of the theory described above to improve the design of markets in which such search takes place. The prospect of designing or significantly influencing these markets is not remote. Consider the design of a large scale Internet website like `autotrader.com`. The listings for cars that users see are signals, and they may be unsure of a car's true worth. `autotrader` can partner with a provider of reports like Carfax, to make it easy for users to look up a car's worth. In order to be general, let us refer to `autotrader` as "the market" (or in some instances as "the market designer") and Carfax as the expert. The market wants to attract customers to it, rather than to rival markets. The best way of doing this is to provide customers with a high value shopping experience. The expert wishes to maximize its profits. Since the market and the expert both have significant power, it is reasonable to imagine them coming up with different models of the kinds of relationships they may have. It is notable that while we are thinking about private markets here, this entire discussion is equally relevant to a big player like the government as market designer, and independent providers of expert services.

In order to provide customers with the highest value shopping experience, the market may choose to subsidize the cost of expert services. A typical problem with subsidization is that it often decreases social welfare because the true cost of whatever is being subsidized is hidden from the consumer, leading to overconsumption of the resource. In this instance, however, the natural existence of many monopolies in expert services, combined with the existence of search frictions, make it quite possible that subsidies will in fact increase social welfare. We show in Section 5 that this is in fact the case for some natural distributions.

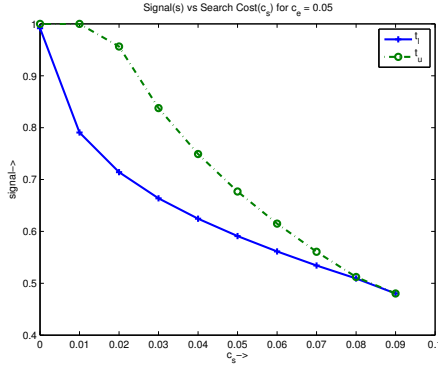
The basic framework of subsidization works as follows. Suppose a monopolist provider of expert services maximizes its profits by setting the querying cost to c_e^* , yielding an expected profit $\pi_e = (c_e^* - d_e)\eta_{c_e^*}$ (this discussion is on a per-consumer basis). The market designer can step in and negotiate a reduction of the fee c_e charged by the expert, for the benefit of the agents. In return for the expert's agreement, the market designer will need to offer a per-consumer payment β to the expert, which fully compensates the expert for the decreased revenue, leaving her total profit the same. Since $c_e' < c_e^*$, $\eta_{c_e'} > \eta_{c_e^*}$ (the consumer queries more often because she has to pay less). The compensation for a requested decrease in the expert's fee from c_e^* to c_e' is thus $\beta = (c_e^* - d_e)\eta_{c_e^*} - (c_e' - d_e)\eta_{c_e'}$. The overall welfare per agent in this case increases by $V_{c_e'} - V_{c_e^*}$, where $V_{c_e'}$ and $V_{c_e^*}$ are the expected value of searchers according to Equation 6-8, when the expert uses a fee c_e' and c_e^* respectively, at a cost β to the market designer. Since the expert is fully compensated for her loss due to the decrease in her fee, the change in the overall social welfare is $V_{c_e'} - V_{c_e^*} - \beta$. Under the new pricing scheme c_e' , and given the subsidy β , the social welfare is given by $W' = V_{c_e'} + \pi_e - \beta$. In the following section we illustrate how such a subsidy β can have a positive change over the social welfare.

An interesting special case to consider is when $d_e = 0$. We can think of this case as "digital services," analogous to digital goods like music MP3s – producing an extra one of these has zero marginal cost. Similarly, producing an extra electronic history of a car, like a Carfax report, can be considered to have zero marginal cost. In this case, there is no societal cost to higher utilization of the expert's services, so subsidy is welfare improving right up to making the service free. These are the cases where it could make sense for the market designer or government to take over offering the service themselves, and making it free, potentially leveraging the increased welfare of consumers by attracting more consumers to their market, or increasing their fees.

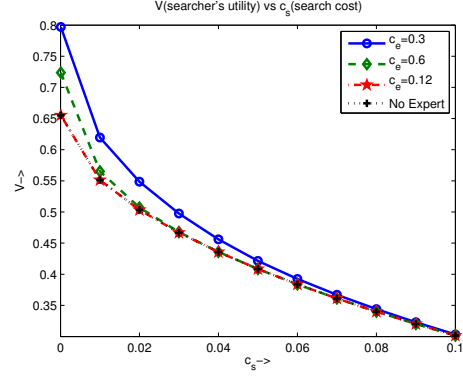
5. A SPECIFIC EXAMPLE

In this section we illustrate the theoretical analysis given in the former section for a particular plausible distribution of signals and values. This case illustrates the general structure of the solutions of the model and demonstrates how interventions by the market designer can increase social welfare.

We consider a case where the signal is an upper bound on the true value. Going back to the used car example, sellers and dealers offering cars for sale usually make cosmetic improvements to the cars in question, and proceed to advertise them in the most appealing manner possible, hiding defects using temporary fixes. Specifically, we assume signals s are



(a) Signal threshold vs c_s



(b) Variation of agent utility w.r.t c_s and c_e

Figure 2: Effect of c_s on the signal thresholds (t_l, t_u) and agent utility V

uniformly distributed on $[0, 1]$, and the conditional density of true values is linear on $[0, s]$. Thus

$$f_s(s) = \begin{cases} 1 & \text{if } 0 < s < 1 \\ 0 & \text{otherwise} \end{cases} \quad f_v(y|s) = \begin{cases} \frac{2y}{s^2} & \text{for } 0 \leq y \leq s \\ 0 & \text{Otherwise} \end{cases}$$

We can substitute in these distributions in Equations 6 through 8 and simplify. From Equation 6:

$$V = -c_s + Vt_l - c_e(t_u - t_l) + \frac{V^3(t_u - t_l)}{3t_u t_l} + \frac{1 - t_l^2}{3}$$

$$V = \frac{2t_l}{3} + \frac{V^3}{3t_l^2} - c_e \quad (\text{from Equation 7})$$

$$c_e = \int_0^V (V - y)f_v(y|t_u) dy = \frac{V^3}{3t_u^2} \quad (\text{from Equation 8})$$

We can find feasible solutions of this system for different parameter values, as long as the condition $t_l < t_u$ holds. Otherwise, when c_e is high enough that querying never makes sense, a single threshold serves as the optimal strategy, as in the case with no expert. In the latter case, we obtain the optimal reservation value to be used by the searcher from Equation 3, yielding $t^* = 1 - \sqrt{3c_s}$.

The other thing to note here is that Equation 8 above is for the case when the support on signal s is unbounded. When there is an upper limit on s i.e $s \leq m$ for some m (as is the case here, where signals are bounded in $[0, 1]$), once t_u reaches m (we never buy without querying), Equation 8 does not hold. Now the system rejects if the signal is below t_l or queries if it is above.

Figure 2(a) illustrates how the reservation values t_l and t_u change as a function of c_s for $c_e = 0.05$. The vertical axis is the interval of signals. As can be seen from the graph, for very small search cost (c_s) values, the searcher never terminates search without querying the expert.¹ Due to the low search cost the searcher is better off only querying the expert when a high signal is received. The expert option is preferred over accepting without querying the expert for those high signals, because if a low value is received from the expert then the cost of finding a new opportunity with a high signal is low. As the search cost c_s increases, there is some

¹When search costs are 0 the problem is ill-defined. The first point on the graph shows an extremely low, but non-zero search cost. In this case $t_u = 1$ and t_l is almost 1, but not exactly, and the expert is again always queried.

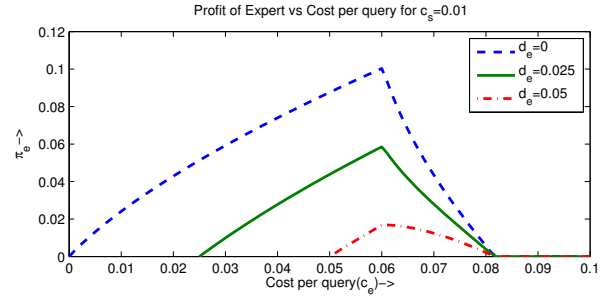


Figure 3: Expert's profit as a function of c_e and d_e for $c_s = 0.01$.

behavior that is not immediately intuitive. The reservation values t_l and t_u become closer to each other until coinciding at $c_s = 0.08$, at which point the expert is never queried anymore. The reason for this is that the overall value of continuing search goes down significantly as c_s increases, therefore the cost of querying the expert becomes a more significant fraction of the total cost, making it comparatively less desirable. This is a good example of the additional complexity of analyzing a system with an expert, because in the static sense the cost of consulting the expert does not change, so the fact that the expert should be consulted less and less frequently is counter-intuitive. Figure 2(b) illustrates the change in the searcher's welfare as a function of the search cost, c_s , for different values of the service fee, c_e , charged by the expert. As expected, the searcher's welfare is better with the expert option than without, and the smaller the fee charged by the expert, the better the searcher's welfare.

Expected number of queries.

We can find the expected number of queries in this case by using our knowledge of the uniform distribution and the noise distribution in Equations 11-15, yielding

$$A = V^2\left(\frac{1}{t_l} - \frac{1}{t_u}\right); \quad B = t_l; \quad C = (1 - t_u);$$

$$D = t_u - t_l - V^2\left(\frac{1}{t_l} - \frac{1}{t_u}\right)$$

which give the final expressions:

$$\eta_{c_e} = \frac{t_l t_u (t_u - t_l)}{t_u t_l^2 - t_l V^2 - t_u t_l + t_u V^2}; \quad \eta_{c_s} = \frac{1}{1 - t_l - V^2\left(\frac{1}{t_l} - \frac{1}{t_u}\right)}$$

The monopolist expert’s optimal strategy.

Using the above derivations, it is now easy to calculate numerically the value of c_e^* that maximizes π_e , trading off decreasing number of queries η and increasing revenue per query c_e . Figure 3 shows examples of the graph of expected profit for the expert as a function of the expert’s fee, c_e , for different values of d_e , the marginal cost to the expert of producing an extra “expert report”, for $c_s = 0.01$.

Subsidizing the expert.

As discussed above, the market designer or the government can guarantee the reduction of the expert’s charge from c_e^* to c_e' , keeping π_e constant, by paying a per-consumer subsidy β to the expert. Figure 4 shows the improvement in social welfare, denoted $\delta(W)$, as a function of the subsidy paid to the expert, β , for various d_e values (where $c_s = 0.01$).

From the graphs, we do indeed find that subsidization can lead to substantial increases in social welfare, even when there is a significant marginal cost of producing an expert report. While this could be from a reduction in search and query costs or an increase in the expected value of the opportunity finally taken, the data in Table 1 indicates that the latter explanation is the dominant factor in this case. It is also worth noting that social welfare is maximized at the point where the searcher pays exactly d_e per query, thus fully internalizing the cost to the expert of producing the extra report. If the searcher had to pay less, it would lead to inefficient overconsumption of expert services, whereas if she had to pay more, the expected decline in the value she receives from participating in the search process would outweigh the savings to the market designer or government from having to pay less subsidy.

6. RELATED WORK

The autonomous agents literature has often considered the problem of search which incurs a cost [2, 9, 10]. The underlying foundation for such analysis is search theory [4, 15], and in particular, its one-sided branch which considers an individual sequentially reviewing different opportunities from which she can choose only one. The search incurs a cost and the individual is interested in minimizing expected cost or maximizing expected utility ([13, 7, 16], and references therein). To the best of our knowledge, none of the one-sided search literature in either search theory or multi-agent systems, has considered the resulting market dynamics when observations are not accurate and more accurate information can be purchased from a self-interested agent.

Relaxation of the perfect signals assumption is typically found in models of two-sided search [1], including marriage or dating markets [3] and markets with interviewing [11]. The literature has not to this point focused on the decision problems faced by self-interested knowledge brokers, or how their presence affects the market.

In terms of market interventions, the two-sided search literature has considered the impact of search frictions on labor markets (the 2010 Nobel Prize in Economics was awarded for this work [15, 4]). One classic regulatory intervention in these models is the introduction of a minimum wage, which can be shown to be welfare increasing in many contexts [8], but we are unaware of any work on subsidizing providers of expert knowledge, as we discuss here.

7. DISCUSSION AND CONCLUSIONS

The power of modeling markets using search theory is well established in the literature on economics and social science [12; 1, *inter alia*]. It has led to breakthroughs in understanding many domains, ranging from basic bilateral trade [17] to labor markets [15]. While knowledge has always been an economically valuable commodity, its role continues to grow in the Internet age. The ubiquity of electronic records and communications means there is an increasing role for knowledge brokers in today’s marketplaces. For example, it is now feasible for agencies like Carfax to collect the available records of every recorded accident, insurance claim, oil change, inspection, and so on for every car. The presence of such knowledge brokers necessitate that we take them into account in modeling the search process of consumers.

This paper takes the first step in this direction. We introduce a search model in which agents receive noisy signals of the true value of an object, and can pay an expert to reveal more information. We show that, for a natural and general class of distributions, the searcher’s optimal strategy is a “double reservation” strategy, where she maintains two thresholds, an upper and a lower one. When she receives a signal below her lower threshold, she rejects it immediately. Similarly, when she receives a signal above her upper threshold, she accepts it immediately. Only when the signal is between the thresholds does she consult the expert, determining whether to continue searching or accept the offer based on the information revealed by the expert.

In such models, there is scope for an authority like a market designer or regulator to improve social welfare by subsidizing the cost of querying the expert. The benefit to the searcher of having less friction in the process could potentially more than offset the cost to the authority. The downside would be that if the authority provided too much subsidy, this could lead to inefficient overconsumption of costly (to produce) expert services. By solving the model for a natural combination of the distribution of signals and the conditional distribution of the true value given the signal, we can analyze such questions more specifically. We show that in our example, subsidies can in fact be welfare-enhancing, and, in fact, social welfare is maximized when the searcher has to pay exactly the marginal production cost of expert services. Both the model and our results are significant for designers of markets in which consumers will search and the need for expert services will arise naturally (like an Internet marketplace for used cars), because by enhancing social welfare, the market designer can take market share away from competitors, or perhaps charge higher commissions, because it is offering a better marketplace for consumers.

There are several directions for future research, from both the expert’s perspective and the market-designer’s. An interesting problem for the monopolist expert is the optimal pricing of bundles of queries, where agents must purchase a bundle, instead of individual reports. More realistic modeling of “startup costs” and hence the average “supply curve” of expert services (instead of the marginal cost considered here) may also explain a richer range of behaviors. The existence of the expert has ramifications beyond one-sided search, our focus in this paper. For example, in two-sided search markets like labor markets, there may be different types of experts: those who conduct background checks, for example, or providers who run independent testing services to vet potential employees. What are their incentives, and

d_e	Without Subsidy						With Subsidy						
	c_e	$\eta_{c_e} c_e$	$c_s \eta_s$	Worth	V	W	subsidy	c_e	$\eta_{c_e} c_e$	$c_s \eta_s$	Worth	V	W
0	0.06	0.10	0.08	0.7472	0.5653	0.6656	0.1003	0	0	0.1036	0.7928	0.6893	0.6893
0.025	0.06	0.10	0.08	0.7472	0.5653	0.6238	0.0585	0.025	0.05	0.09	0.7712	0.6295	0.6295
0.05	0.061	0.09	0.08	0.7365	0.5637	0.5806	0.0169	0.05	0.09	0.08	0.7536	0.5824	0.5824

Table 1: The different components of social welfare with and without subsidy for $c_s = 0.01$. “Worth” is the expected value of the opportunity eventually picked. Initially the decrease in query cost contributes more to the increase in social welfare, but as d_e increases, this contribution becomes less significant. Note that the first two columns in the case without subsidy are similar because the profit-maximizing c_e is the same and the searcher’s cost depends only on value of selected c_e , not d_e .

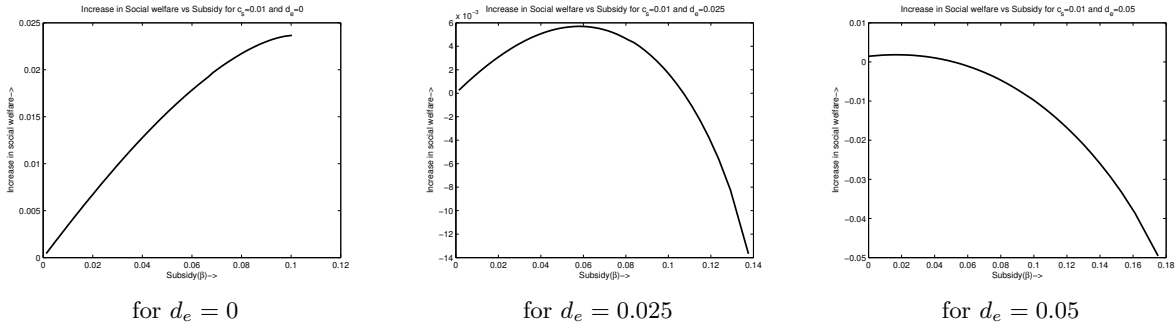


Figure 4: Increase in social welfare vs subsidy. When there is no marginal cost ($d_e = 0$), it is better for the market designer to make the service available for free but when there is some marginal cost involved, then increase in social welfare is a concave peaked at marginal cost.

how do these affect two-sided search markets? From the market designer’s point of view, new alternatives to subsidization as a means for improving social welfare can be explored, e.g., inducing competition, or provision of expert services by the market designer herself (e.g., a government takes over the role of providing expert services).

8. ACKNOWLEDGMENTS

We are grateful to the US-Israel BSF for supporting this research under Grant 2008404. Das also acknowledges support from an NSF CAREER award (0952918), and Sarne is partially supported by ISF grants 1401/09.

9. REFERENCES

- [1] K. Burdett and R. Wright. Two-sided search with nontransferable utility. *Review of Economic Dynamics*, 1:220–245, 1998.
- [2] S. Choi and J. Liu. Optimal time-constrained trading strategies for autonomous agents. In *Proc. MAMA*, 2000.
- [3] S. Das and E. Kamenica. Two-sided bandits and the dating market. In *Proc. IJCAI*, 2005.
- [4] P. Diamond. Aggregate demand management in search equilibrium. *The Journal of Political Economy*, 90(5):881–894, 1982.
- [5] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.
- [6] J. Gilbert and F. Mosteller. Recognizing the maximum of a sequence. *Journal of the American Statistical Association*, 61:35–73, 1966.
- [7] A. Grosfeld-Nir, D. Sarne, and I. Spiegler. Modeling the search for the least costly opportunity. *European Journal of Operational Research*, 197(2):667–674, 2009.
- [8] L. Ho. Wage subsidies as a labour market policy tool. *Policy Sciences*, 33(1):89–100, 2000.
- [9] J. Kephart and A. Greenwald. Shopbot economics. *JAAMAS*, 5(3):255–287, 2002.
- [10] J. O. Kephart, J. E. Hanson, and A. R. Greenwald. Dynamic pricing by software agents. *Computer Networks*, 32:731–752, 2000.
- [11] R. Lee and M. Schwarz. Interviewing in two-sided matching markets. *NBER Working Paper*, 2009.
- [12] S. Lippman and J. McCall. The economics of job search: A survey. *Economic Inquiry*, 14:155–189, 1976.
- [13] J. McMillan and M. Rothschild. Search. In R. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, pages 905–927. 1994.
- [14] P. Milgrom. Good news and bad news: Representation theorems and applications. *The Bell Journal of Economics*, pages 380–391, 1981.
- [15] D. Mortensen and C. Pissarides. New developments in models of search in the labor market. *Handbook of labor economics*, 3:2567–2627, 1999.
- [16] M. Rothschild. Searching for the lowest price when the distribution of prices is unknown. *Journal of Political Economy*, 82:689–711, 1961.
- [17] A. Rubinstein and A. Wolinsky. Middlemen. *The Quarterly Journal of Economics*, 102(3):581–593, 1987.
- [18] M. Weitzman. Optimal search for the best alternative. *Econometrica: Journal of the Econometric Society*, 47(3):641–654, 1979.
- [19] R. Wright. Job search and cyclical unemployment. *The Journal of Political Economy*, 94(1):38–55, 1986.

Using Aspiration Adaptation Theory to Improve Learning*

Avi Rosenfeld¹ and Sarit Kraus²

¹Department of Industrial Engineering
Jerusalem College of Technology, Jerusalem, Israel 91160

²Department of Computer Science
Bar-Ilan University, Ramat-Gan, Israel 92500
rosenfa@jct.ac.il, sarit@cs.biu.ac.il

ABSTRACT

Creating agents that properly simulate and interact with people is critical for many applications. Towards creating these agents, models are needed that quickly and accurately predict how people behave in a variety of domains and problems. This paper explores how one bounded rationality theory, Aspiration Adaptation Theory (AAT), can be used to aid in this task. We extensively studied two types of problems – a relatively simple optimization problem and two complex negotiation problems. We compared the predictive capabilities of traditional learning methods with those where we added key elements of AAT and other optimal and bounded rationality models. Within the extensive empirical studies we conducted, we found that machine learning models combined with AAT were most effective in quickly and accurately predicting people’s behavior.

Categories and Subject Descriptors

I.2.11 [Artificial Intelligence]: Distributed Artificial Intelligence

General Terms

Experimentation

Keywords

bounded rationality, cognitive models, agent learning

1. INTRODUCTION

The challenge of creating agents that effectively simulate and interact with people is of utmost importance in many systems [7, 8, 9, 16]. These agents form the backbone of many mixed human-agent systems such as entertainment domains [7], Interactive Tutoring Systems [9], and mixed human-agent trading environments [8]. In order to effectively interact with people, the agents have to understand and predict people’s behavior. To date, these agents have often been created based on the perspectives of unbounded

rationality including expected utility, game theory, Bayesian models, or Markov Decision Processes (MDP) [10, 15].

While these models are mathematically elegant and have proven effective in some situations [10, 15], several fundamental obstacles exist in applying them to many real-world applications. First, previous research in experimental economics and cognitive psychology has shown that human decision makers often do not adhere to fully rational behavior. For example, Kahneman and Tversky [3] have shown that individuals often deviate from optimal behavior as prescribed by Expected Utility Theory. Second, decision makers often lack complete information and thus do not necessarily know the quantitative structure of the environment in which they act. Thus, even assuming that people act rationally, they cannot always compute the optimal solution for a given problem, as they lack facts required to arrive at this decision. Finally, even if people wish to act rationally and have complete information about a given problem, it may still be impossible for them to compute the optimal solution. Previous research has found that many classes of real-world problems exist for which finding the optimal sequence of actions is of intractable computational complexity [12]. Thus, even in the best of circumstances, expecting people to behave optimally based on full rationality is unrealistic.

We posit that models based on Bounded Rationality hold the most promise to best predict people’s behavior. This research direction, initiated by Simon [17], assumes that people – except in the simplest of situations – lack the cognitive and computational capabilities to find optimal solutions. Instead they proceed by searching for non-optimal alternatives to fulfill their goals. Simon coined the term “satisfice” to capture that bounded decision makers seek “good enough” solutions and not optimal ones. In this tradition, Sauer mann and Selten proposed a framework called Aspiration Adaptation Theory (AAT) [16] as a boundedly rational model of decision making.

This paper’s major contribution is support for the claim that learning models that aim to predict people’s behavior should be based on bounded rationality models, and specifically AAT. To empirically support this claim, we studied two types of problems – one relatively simple optimizing problem and two complex negotiation problems. Within the optimization problem, we found that traditional machine learning methods such as decision trees were successful in discovering the strategies most people used. We note that these strategies were consistent with AAT and were not representative of other bounded theories or the problem’s optimal policy. Furthermore, we found that an AAT based predic-

*This research is based on work supported in part by the U.S. Army Research Laboratory and the U.S. Army Research Office grant # W911NF-08-1-0144, NSF grant # 0705587 and the Israel Ministry of Science and Technology grant # 3-6797. Sarit Kraus is also affiliated with UMIACS.

Cite as: Using Aspiration Adaptation Theory to Improve Learning, Avi Rosenfeld and Sarit Kraus, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 423-430.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

tion policy was extremely accurate, even when trained with very limited data. Second, in the negotiation domains we studied, traditional learning algorithms such as those in the Weka learning package [21] were unable to effectively predict how people would behave due to the inherent problem complexity. Nonetheless, by adding statistical information about people’s bounded AAT strategies, we were able to significantly improve models’ prediction accuracy beyond the base algorithm.

2. RELATED WORK

Many multi-agent researchers have found that classical models based on rigid models of expected utility often are not effective in domains they studied. [1, 4, 5, 6, 8, 14]. Consequently, a key research question is what should be used in place of these models. For example, designers of automated negotiation agents found that when the automated agents follow their equilibrium strategies, the human negotiators who negotiate with them become frustrated as the equilibrium strategy requires the automated agent to repeatedly propose the same offer. Thus, negotiation sessions often ended with no agreement [4]. Similarly, Lin et al. [6], created an equilibrium agent for the negotiation domains studied in this paper. They also found that in many situations the equilibrium agent failed to reach any agreement when playing with people [6]. Thus, both the KB and QO agents they created also intentionally steered away from the equilibrium strategies [6]. In fact, a survey article of automated negotiators [5] found that none of the reviewed agents implemented equilibrium strategies. They attributed this to the assumption that people avoid equilibrium strategies in complex environments due to their bounded nature.

However, once classical rationality models have been rejected, the question then becomes what model to use instead. Towards this goal, previous works typically model people’s preferences based on historical information. For example, Gal and Pfeffer use a statistical approach to learn which bid a person will select from a known number of possibilities [1]. Pardoe and Stone used machine learning techniques to create a trading agent to predict the likelihood that a person would accept a given bid price [11]. However, a key methodology difference exists between these and works similar to ours. While previous works typically used existing machine learning algorithms, our goal is to find a general model of bounded rationality and use it to better learn people’s actions. This important difference impacts two key research issues. First, by using only generalized bounded theories, we can prove or disprove the applicability of psychological or economic bounded models in new problem domains. Possibly more importantly, and as we demonstrate in this paper, once a generalized theory is found to be accurate, it can be applied to new problems in order to further improve existing learning algorithms’ accuracy above traditionally used machine learning methods.

Previously, we studied the applicability of bounded rationality models, and even AAT [14]. However, that study focused on validating if AAT was relevant, and not how it could be used for improving learning accuracy. Additionally, that work used human judges to decide if AAT was being used, and thus leaves open concerns regarding possible human bias. This paper is the first that uses classic decision tree models [13] to decide what decision model, bounded or not, is used and how to use these theories for improved learning.

3. ASPIRATION ADAPTATION THEORY

Aspiration Adaptation Theory (AAT) was proposed by Selten as a general economic model for how people make certain economic decisions without any need for expected utility functions [16]. AAT was originally formulated to model how people make decisions where utility functions cannot be constructed. For example, assume you need to relocate and choose a new house to live in. There are many factors that you need to consider, such as the price of each possible house, the distance from your work, the neighborhood and neighbors, and the schools in the area. How do you decide which house to buy? While in theory utility based models could be used, many of us do not create rigid formulas involving numerical values to weigh trade-offs between each of these search parameters.

AAT provides an alternative to utility theory for how decisions can be made in this and other problems. First, m goal variables are sorted in order of priority, or their *urgency*. Accordingly, the order of G_1, \dots, G_m refers to goals’ urgency, or the priority by which a solution for the goal variables is attempted. Each of the goal variables has a desired value, or its *aspiration level*, that the agent sets for the current period. This desired value is not necessarily the optimal one, and the agent may consider the variable “solved” even if it finds a sub-optimal, but yet sufficiently desired value. The agent’s search starts with an initial aspiration level and is governed by its *local procedural preferences*. The local procedural preferences prescribe which aspiration level is most urgently adapted upward if possible, second most urgently adapted upward if possible, etc. and which partial aspiration level is *retreated from* or adapted downward if the current aspiration level is not feasible. Here, all variables except for the goal variable being addressed are assigned values based on *ceteris paribus*, or all other goals being equal a better value is preferred to a worse one.

The search procedure as described by AAT is different from traditional search methods such as Hill-climbing or machine learning methods such as Gradient Descent techniques in two aspects. First, within traditional learning or search, optimal values for all variables are sought for simultaneously [15]. In contrast, within AAT only one goal is attempted to be satisfied at a time. Second, in AAT, the focus is on “satisficing” goal values based on their *aspiration levels*. This approach makes no attempt to find optimal values beyond these “good enough” values – something machine learning methods do search for.

It is important to note that two key differences exist between classic AAT, and how we apply AAT within this study. First, AAT assumes that the m goal variables used to solve \mathcal{G} are incomparable as no utility function is possible to connect goal variables. For example, in buying a house the goal variables for location, price and size are likely to be incomparable with no evident utility function to compare them. In this paper, we consider simpler problems where a concrete function between \mathcal{G} and the m goal variables clearly exists. Nonetheless, we hypothesize that people will not attempt to calculate \mathcal{G} due to their bounded nature. This represents a significant generalization to AAT’s theory and its relevance even within domains that contain concrete, albeit difficult to quantify, utility functions. Second, AAT is based on the premise that the person’s search will be based on an *aspiration scale* which sorts the m goal variables and attempts to satisfice values for these goals. As we consider optimization

and negotiation problems where utility can be calculated, it is more natural for people to consider optimizing the instrument variables that constitute the basis of these goals rather than the more abstract general goal variables. This difference again represents a significant generalization of AAT.

We recognize that AAT is not the only possible model that can predict human decisions. In both of the problems we studied, optimal search methods or negotiation strategies could have been used. Additionally, other bounded strategies other than AAT are possible. Psychological models of bounded rationality have suggested that people use domain specific biases or heuristics to solve problems sub-optimally [2, 3]. Following the psychological approaches, simple predefined or greedy heuristics could be used. In the optimizing domains, predefined values could have been used instead of attempting to solve the problems. Within the negotiation domains simple compromise heuristics could have been used. Possibilities of such heuristics include always countering the middle position between the previous offer of both sides or offering the middle position between all previous offers of both sides. These types of approaches would be consistent with basic implementations of Gigerenzer and Goldstein’s fast and frugal heuristics [2] and involve using simplistic preset values that are seen as “good enough”.

4. EXPERIMENT SETUP

We studied how people solved two types of problems – a relatively simple optimization problem and more complex negotiation problems. We used the optimization problem as a baseline as we believe it would be easier to understand the decision making models used by people in this problem. By focusing on more complex problems as well, we show the applicability of our findings to real-world applications.

We specifically studied negotiation problems as various real-world tasks are based on negotiation capabilities. These can be as simple and ordinary as haggling over a price in the market or deciding what television show to watch. Negotiation issues can also involve issues where millions of lives are at stake, such as interstate disputes [20]. The use of simulation and role-playing is common for training people in negotiations (e.g., the Interactive Computer-Assisted Negotiation Support system (ICANS)) [18]. These simulations can be used in conjunction with people to alleviate some of the efforts required of people during negotiations and also assist people that are less qualified in the negotiation process. Additionally, there may be situations in which simulations can even replace human training procedures.

4.1 Optimization Domain

In the first optimization problem, we consider a problem where a person must minimize the price in buying a commodity (a television) given the following constraints. Assume a person must personally visit stores in order to observe the posted price of the commodity. However, some cost exists from visiting additional stores. We assume this cost is due to factors such as an opportunity cost with continuing the search instead of working at a job with a known hourly wage. For any given discrete time period, the person must decide if she wishes to terminate the search. At this point, we assume she can buy the commodity from any of the visited stores without incurring an additional cost. The goal of the agent is to minimize the overall cost of the process which is the sum of the product cost and the aggregated

search cost.

From a strategic point of view, the game is played under a time constraint. An optimal solution to this problem can be found as an instance of the Pandora’s problem [19] resulting in a stationary threshold below which the search should be terminated. Formally, we can describe this problem as follows: We assume that there is a finite timeline $\mathcal{T} = \{1, 2, \dots, k\}$. In each time step t , $t \leq k$, the agent observes a cost and needs to decide whether to end the search. All of the observed costs, regardless of the time step, are drawn from the same distribution. We denote c_t as the lowest price the agent observed up to and including the time period t (i.e., $c_t \leq c_{t-1}$). At the end of the game the agent’s cost is $cost(t, c_t) = c_t + \lambda * t$, $\lambda > 0$. The agent’s goal is to minimize this cost. As has been previously proven, the optimal strategy in such domains is as follows: exists \bar{c} such that if $c_t \leq \bar{c}$ the agent should stop the search [19].

Intuitively, it seems strange that the decision as to whether the agent should stop the search does not depend on how much time is left, i.e., \bar{c} does not depend on $k - t$. However, the reason for this is as follows. If the agent’s overall expected benefit from continuing the search (i.e., the reduction in price that it will obtain) is lower than the overall cost due to the added search time, the agent clearly should not continue the search. Furthermore, it was proven that it is enough for the agent to consider only the next time period, i.e., it should stop the search if and only if the expected reduction in the price in the next time period is less than the cost of continuing one time period (λ) [19].

In our implementation, the prices are distributed normally with a mean μ and a standard deviation σ . We denote by x the price for which the expected reduction in the price for one time period is equal to λ . For a given price p the benefit is $x - p$ and the probability¹ for p is

$$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{p-\mu}{\sigma}\right)^2}$$

Given these definitions we must generally solve:

$$\int_0^x (x - p) \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{p-\mu}{\sigma}\right)^2} dp = \lambda$$

In our specific implementation, $\mu = 1000$, $\sigma = 200$ and $\lambda = 15$. Thus we specifically solve,

$$\int_0^x (x - p) \frac{1}{200\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{p-1000}{200}\right)^2} dp = 15$$

Solving this equations yields a solution of $x = 789$.

4.2 Negotiation problems

We also studied two previously defined negotiation domains [6] and focused on how people came to agreements with other people in these problems. The goal of these problems was to negotiate as high a utility as possible given a known weight of all issues. To reach an agreement, people sent offers through a web interface which facilitated their choosing the different values that constitute an offer. This offer was then sent to the other person in plain English. A time effect existed that assigned a time cost which influences

¹In the domain, when a negative price was drawn, we drew a new price. Since the probability of such an event is extremely small, we did not consider it in our analysis.

the utility of each player as time passed (there can be different time costs for each player). The time effect was either negative or positive. If no agreement is reached by the end of the final turn then a status quo agreement was implemented resulting in a status quo value for each player. Each player could also quit the negotiation session at any given time if he/she decided that the negotiation session is not proceeding in a favorable way. This resulted in the implementation of an opt out outcome. During each phase of the negotiation session, the instructions and parameters subject to negotiation were accessible to the players. The players were also aware of the current turn and time left until the end of the turn and until the negotiation session terminates. The history of past sessions was also easily accessible. When receiving an offer the player can choose whether to accept or reject it, or make a counter-offer.

4.2.1 Employer / Employee Negotiation Domain

In the first problem, we consider a negotiation session that takes place after a successful job interview between an employer and a job candidate. In the negotiation session both the employer and the job candidate wish to formalize the hiring terms and conditions of the applicant. Below are the issues under negotiation: The **Salary** issue dictates the total net salary the applicant will receive per month. The possible values are (a) \$7,000, (b) \$12,000, or (c) \$20,000. The **Job Description** issue describes the job description and responsibilities given to the job applicant. The job description has an effect on the advancement of the candidate in his/her work place and his/her prestige. The possible values are (a) QA, (b) Programmer, (c) Team Manager, or (d) Project Manager. In addition to the base salary, other job benefits may also be negotiated. The **Car Benefits** issue revolves around the possibility that the company will provide a company car for use by the employee with possible values being (a) a leased company car or (b) no leased car. Pension benefits must also be negotiated and set as percentage of the work's salary. The possible value for the **Pension benefits** are (a) 0%, (b) 10%, or (c) 20%. The **Promotion possibilities** issue describes the commitment by the employer regarding the fast track for promotion for the job candidate. The possible values are (a) fast promotion track (2 years), or (b) slow promotion track (4 years). The **Working hours** issue describes the number of working hours required by the employee per day (not including over-time). The possible values are (a) 8 hours, (b) 9 hours, or (c) 10 hours. In total, this scenario allows for a total of 1,296 possible agreements ($3 \times 4 \times 12 \times 3 \times 3 = 1296$)².

Each turn in the negotiation simulation is equivalent to two minutes of an actual negotiation session, and the total negotiation session is limited to 28 minutes. If the sides do not reach an agreement by the end of the allocated time, the job interview ends with the candidate being hired with a standard contract, which cannot be renegotiated during the first year. This outcome is modeled for both agents as the status quo outcome.

During negotiation, each side can also opt-out of the negotiation session if it feels that the prospects of reaching an agreement with the opponent are slim or if they feel it is no longer able to negotiate a fair deal. If the employer

²Note that it is possible for there to be no agreement for many of these parameters. In these case, we considered the negotiation to have failed.

opts out then she will incur an expense due to the lost time and work from the potential employee. As we assume the employer will be required to postpone the project for which the candidate was interviewing, this cost can be considerable. Conversely, the employee also has some leverage. If the employee accepts too little, he is likely to find better work elsewhere. However, the employee also loses the time and salary from lost wages in continuing their job search. Additionally, opting-out will make it very difficult for him to find another job, as the employer will spread his/her negative impression of the candidate to other CEOs of large companies.

Time also has an impact on the interaction. As time advances the candidate's utility decreases, as the employer's good impression has of the job candidate decreases. The employer's utility also decreases as the candidate becomes less motivated to work for the company.

4.2.2 Political Dispute Negotiation Domain

The second negotiation is based on a scenario where England and Zimbabwe attempt to reach an agreement evolving from the World Health Organization's Framework Convention on Tobacco Control, the world's first public health treaty. The principal goal of the convention is "to protect present and future generations from the devastating health, social, environmental and economic consequences of tobacco consumption and exposure to tobacco smoke." The leaders of both countries are about to meet at a long scheduled summit and must reach an agreement on the following issues: The **Creation of a Global Tobacco Fund** issue describes the total amount to be deposited into the Global Tobacco Fund to aid countries seeking to rid themselves of economic dependence on tobacco production. This issue has an impact on the budget of England and on the effectiveness of short-range and long-range economic benefits for Zimbabwe. The possible values are (a) \$10 billion, (b) \$50 billion, or (c) \$100 billion. The **Impact on other aid programs** issue affects the net cost to England and the overall benefit for Zimbabwe. If other aid programs are reduced, the economic difficulties for Zimbabwe will increase. The possible values are (a) no reduction, (b) reduction equal to half of the Global Tobacco Fund, or (c) reduction equal to the size of the Global Tobacco Fund. Both Zimbabwe and England must negotiate **Trade Issues**. Countries can use restrictive trade barriers such as tariffs (taxes on imports from the other country) or they can liberalize their trade policy by increasing imports from the other party. There are both benefits and costs involved in these policies: tariffs may increase revenue in the short run but lead to higher prices for consumers and possible retaliation by affected countries over the long run. Increasing imports can cause problems for domestic industries. But it can also lead to lower consumer costs and improved welfare. For both Zimbabwe and England possible values for this are: (a) reducing tariffs on imports or (b) increasing tariffs on imports. The **Forum to Study Long-Term Health Issues** issue revolves around the scope of a forum to explore comparable arrangements for other long-term health issues. This issue relates to the precedent that may be set by the Global Tobacco Fund. If the fund is established, Zimbabwe will be highly motivated to apply the same approach to other global health agreements. This would be very costly to England. The possible values are (a) creation of a fund, (b) creation of a committee to discuss the cre-

ation of a fund, or (c) creation of a committee to develop an agenda for future discussions. Consequently, a total of 576 possible agreements exist ($4 \times 4 \times 3 \times 3 \times 4 = 576$)³.

5. EXPERIMENTAL RESULTS

In this section we detail the shortcomings of using optimal models to predict people’s behavior, and how AAT, and not other bounded methods, should be used instead. In general, we found that in the optimization and negotiation problems we studied, key elements of AAT were present in decisions people made. When sufficient data was present, as we found was the case in the relatively simple optimization problem, clear strategies consistent with AAT were learned by standard machine learning algorithms such as C4.5 [13]. We found that estimates of people’s aspiration scales in this problem were useful for formulating a very accurate prediction model even without an extended learning period and with only sparse data. Within the more complicated negotiation problems, we found that adding statistical information about people’s typically aspiration scales was critical for improving the prediction models as using both the equilibrium strategies and traditional learning methods yielded an extremely poor predictor of how people would act.

5.1 AAT to Predict Optimization Decisions

Our first goal was to use machine learning techniques to determine if optimal policies, fast and frugal heuristics [2] or AAT best predict people’s optimizing decisions. Recall from Section 3.1 that an optimal policy exists based on price alone – buy if the price in the current store is less than 789. Thus classical expected utility theory would predict that people would similarly buy the commodity at this price. Assuming people used fast and frugal search heuristics, we would expect them to formulate simple strategies involving only one variable (e.g. search until price $< X$, or visit Y stores and buy in the cheapest store). However, using an AAT based model for prediction would assume some type of combination strategy exists where one variable is first searched for, but then retreated from assuming that value could not be satisfied. For example, a person might initially search for a price less than 650, but will settle on even a higher price (e.g. the lowest found so far) after unsuccessfully finding this price after 5 stores.

We did in fact find strong support that AAT best predicted when people would buy the commodity. To study this point, we studied the log files taken from 41 people and how they chose to buy the commodity. Each person was presented with a simulated implementation of the problem described in Section 3.1⁴. Every interaction with the simulation randomized the values for the commodity as described in Section 4.1, and all people were told to interact with this simulation until they formulated a clear policy for how they would decide to buy the commodity. After this point, we then logged a minimum of 20 additional interactions (average 25.56) where each interaction ended in a decision to buy the commodity in a certain store. Our goal was to predict where each person would end his search process.

To obtain a prediction model without bias, we first sep-

arated all logged interactions for each of the 41 people, entered them as input for the Weka machine learning package [21], and applied the C4.5 decision tree classification algorithm as implemented by Weka to decide if each person’s behavior was consistent with AAT. We chose this learning algorithm, as opposed to others such as Bayes or Neural Nets, as the output from the C4.5 algorithm would not just predict the person’s behavior, but also provide the rules by which the classifier operates. We could then judge if these rules were consistent with the optimal rule (e.g. purchase if price < 789) or if the classifier represented a fast and frugal rule with only one rule (e.g. buy after 4 stores). Weka found that 30 of the 41 strategies had decision rules based on price and store combinations (e.g. buy immediately if the price is less than 700, otherwise visit 5 stores and buy in the cheapest one) which are clearly non-optimal and not frugal. Instead, this decision process can be viewed as a classic example of the urgency and retreat process within AAT. In these strategies, a certain price threshold is desired (highest urgency) but retreated from if believed to be unattainable. Of the remaining 11 rules, 10 rules were found to be based on the price variable alone, with one based on the number of stores. While none of these 11 rules were optimal, they may be viewed as fast and frugal heuristics. Thus, the C4.5 learning algorithm found that the majority of strategies (30 of 41 or 73%) were consistent with urgency and retreat concepts of AAT, while the minority (27%) of the strategies were an alternate bounded rationality model – namely simpler fast and frugal heuristics. None of the strategies were found to be optimal. This result provides strong support to the claim that AAT, and not optimal or fast and frugal heuristics best predict people’s behavior. Also note that these results are consistent with our previous work [14] where human judges were used instead of machine learning techniques.

Next, we wished to create a general prediction model and check how AAT might be used to improve such a model. To create and test such a model, we combined all 41 people’s logged interactions to create a total of nearly 5000 instances where people either decided to buy the commodity or to continue their search. Using this data, we constructed two baseline models, found in Figure 1. One baseline is a *Naive* model that classifies all decisions based on the majority class, here assuming people will always continue the search. As people didn’t typically buy the commodity right away, the majority decision is to continue the search and thus 78.56% of all decisions are of this type (see Column 1). A second baseline class is a *Learning* decision tree model constructed which was trained using Weka’s C4.5 classifier on the combined data and tested with cross-validation (Column 2). It is interesting to note that this decision tree was also consistent with AAT, but this result is not surprising as most individual logs were consistent with AAT as well.

We then compared these baselines to models which contained adding information about people’s AAT preferences. Column 3 of Figure 1 is based on a learned C4.5 model, but added information about people’s average AAT preferences (e.g. buy immediately if the price is less than 750, otherwise settle on the best price after visiting a total of 3 stores). While this model is slightly better than the learned baseline, this model was not significantly more accurate. We hypothesized this is due to learned baseline being already based on AAT. Thus, by adding additional information we did not significantly aid the C4.5 to improve its accuracy.

³We again consider the negotiation as having failed if no agreement is reached on all issues.

⁴A web interface for this problem can be seen at: <http://www.jct.ac.il/~rosenfa/costSearch.html>

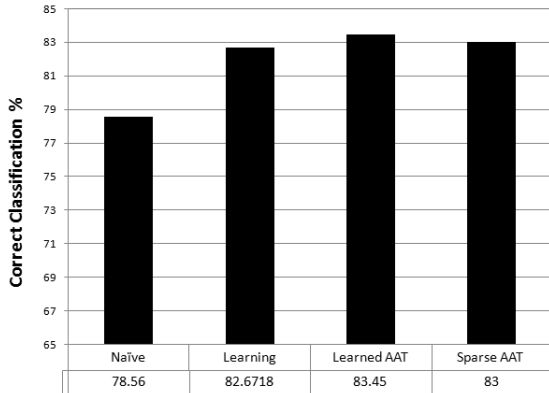


Figure 1: Comparing the Prediction Accuracy between AAT and non-AAT Based Models

We hypothesized that when AAT preferences are clear, such as seems the case in this problem, even sparse data could be used to form an accurate model. In this problem, the most urgent goal variable is the commodity price in the current store. By sampling only a limited number of instances where people stopped the search based on this value, we believe it is possible to approximate the accuracy of the learned model from nearly 5000 logged instances. To support this claim, we formed a learning policy based on the average price used in 50 decisions to buy the commodity and simply averaged the threshold for this parameter (average search stopped at price = 767). The accuracy of this policy is presented in column 4 of Figure 1. Observe that this model is nearly equally accurate to the learned policies. In contrast, creating traditional models with such small amounts of data were not successful and yielded the naive model (Column 1). We also observed that even extremely small samples of only 10 decisions formed similar models. We took 5 such samples and noted that the deviation between these samplings was not great and the lowest prediction accuracy was 81.92%. Thus, we conclude that in relatively basic domains, an accurate learning model can be made even with only limited AAT data based on the most important search parameter(s).

5.2 AAT to Predict Negotiation Decisions

In order to study more complex, real-world problems, we also studied if AAT could be used to better predict people’s negotiation activities in the two problems described in Section 3.2. Recall that in these problems people must negotiate either 5 or 6 parameters. In this section, we study two key issues: 1) Is AAT applicable to negotiation problems? 2) Assuming AAT is applicable, will it facilitate a better prediction model for people’s behavior?

According to AAT, one would expect people to rank the importance of each of the negotiation parameters according to his or her aspiration scale. Assuming people often have the same aspiration scales, we would also see an order where issues are addressed, e.g. certain parameters are typically negotiated first, second, etc. For example, in the employer / employee domain, we might find that negotiations first focus on the salary amount parameter and only then move on to other parameters such as pension or transportation benefits. Our premise is that by understanding these scales, one can

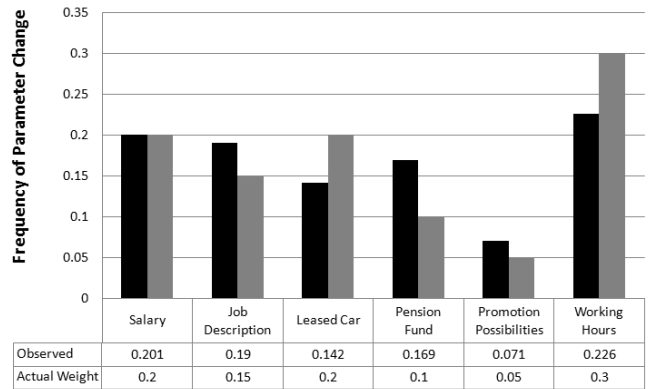


Figure 2: Frequency of Parameter Change in Employer / Employer Negotiation Domain

add this information into traditional models such as C4.5 to more accurately predict what bids people will offer.

We did in fact find that aspiration scales existed whereby certain negotiation parameters were stressed more frequently than others. To study this point, we analyzed how frequently the given parameters were changed, on average, over the course of all collected negotiation interactions. As a baseline, we also considered the actual weights these issues were given [6]. This allowed us to compare if people stressed negotiation issues differently from this baseline value.

Figure 2 presents how frequently each of the 6 parameters in the employer / employee domain were changed. These results were taken from 47 negotiation sessions between people. The first column in this Figure shows the frequency people changed each of these issues (either a raised or lower value). The second value represents the actual weights these issues had. Note that clear aspiration scales existed, and these scales were different from the actual issues weights. Certain parameters, say promotion possibilities, clearly had a lower urgency as these issues were typically not discussed. In comparison, other parameters, such as working hours and salary, were discussed frequently. Furthermore, we observed that even within issues such as working hours and salary that seemed to be equally discussed, the point where they entered in negotiation was often not the same. Often the salary point was discussed first, and only then the number of hours. For example, within the first two negotiation interactions, the salary issues was discussed in 45% of all session, while the number of hours issues was only discussed in 24% of the sessions. Again, this would represent that the salary issue had a higher urgency at the start of negotiations.

Figure 3 presents how frequently each of the 5 parameters in the tobacco trade domain were changed. These results were taken from 56 negotiation sessions between people. Again, certain issues such as the Impact on other Aid and Forum on Other Health Issues were clearly discussed more frequently than issues such as England and Zimbabwe’s Trade Policy. We again noted that certain issues were typically negotiated at different points of sessions. Also note that the aspiration scales here differ greatly from the actual weights of the issues. The actual weight for the Size of Fund parameter was equal to all of issues combined (0.5 of the total weight). Yet, people typically focused on other issues more, such as the Impact on Other Aid and Forum on Other Health Issues parameters. Thus, deriving aspirations from

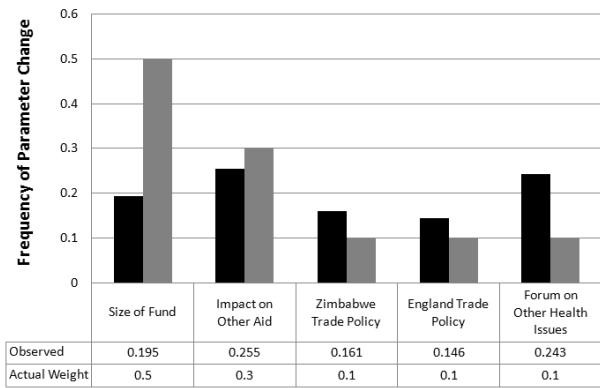


Figure 3: Frequency of Parameter Change in Trade Negotiation Domain

the actual utility weights is not possible.

We then proceeded to study if adding AAT information was helpful in predicting how people will negotiate. In the problems we considered, the parameters to be negotiated could have between 2 and 4 discrete values. Please refer back to Section 3.2 for a description of these parameters and their possible values. The goal of the models was to accurately predict what changes, if any, would be made by a person in the next offer.

In order to study this point we considered several models for both negotiation problems (see Tables 1 and 2). The goal of all of these models was to predict the next value for each parameter. In the learning models, each parameter was training and tested separately through cross-validation, but did have access to the previous values for all parameters. First, we considered the **Majority Rule** model. Given the full log file, this rule assumes that a person would offer the most popular value for any given parameter. For example, in the employer / employee domain, the most popular **title** was “Programmer”. Second, we implemented two models based on the **equilibrium strategy**. This strategy is based on previous work in these problems [6]. However, as the equilibrium strategy will change based on which person is allowed to offer the last bid, we checked both what equilibrium strategies would predict for all parameters. Next, we created a baseline strategy that uses the **C4.5** algorithm to predict the next offer for each parameter. This model used historical information about the previous offer and the current negotiation iteration. Next, we created a **C4.5 with AAT statistical information** prediction model. As we previously demonstrated, each parameter had different urgencies. Thus, we attempted to create a more accurate model by adding information about which parameters were typically raised or lower for any given iteration. Specifically, we added a field with a binary flag value to differentiate between the iterations for which people typically changed a given parameters’ value with a frequency of ≥ 0.5 , and those which were typically not changed and added information would likely not help. This was done to avoid overfitting the AAT statistics for any training / testing pair, and to thus keep the generality of the results. Finally, we created a **C4.5 + Complete Behavior Knowledge** model. This final baseline had knowledge about what the previous offer was, and also added perfect knowledge if the person would revise upwards, downwards, or leave unchanged their previ-

ous offer. In cases where only two options exist, one would expect this baseline to guarantee 100% accuracy. However, when more than 3 values exist for a given parameter, even this model cannot guarantee 100% accuracy. For example, if a previous salary offer was \$7,000 per month and we know the next offer will be higher, we still do not know if it will be raised to \$12,000 or \$20,000. Nonetheless, the goal of this model was to provide an upper bound for how much AAT based information could theoretically help.

Tables 1 and 2 demonstrate the effectiveness of adding AAT information to boost prediction accuracy. The first row of these tables show the parameter to be negotiated and the number of possible values. The second row presents the majority rule baseline. The third and fourth rows present how effective the equilibrium policies were in predicting what people actually offered. Note that both of these policies in both problems fall well below the naive majority baseline. This again demonstrates the ineffectiveness of using equilibrium theoretical policies to predict how people actually behave. The fifth row presents the accuracy of the learned C4.5 model. This model represents the effectiveness of this traditional learning method in predicting each of the parameters. We then added AAT information, and reran the same C4.5 algorithm, the results of which are in the sixth row. Note that in both domains the improvement gained from the AAT information is significant. However, in the Tobacco Trade Domain (Table 2), the prediction improvement is much larger from the base C4.5 algorithm (over a 10% accuracy boost for many parameters), yet falls short of the accuracy in the Employer / Employee Work Domain (Table 1). Also, in both domains, only one parameter did not gain from the added aspiration information. For both of these parameters, few instances existed where people had clear general aspiration changes, preventing any accuracy boost from this approach. Finally, the last line in both domains presents the accuracy of the C4.5 algorithm with complete behavior knowledge, or perfect information about whether a person will retreat from (decrease) a given parameter value, or upwardly revise its aspiration (increase). Note that as expected even complete AAT information could not yield 100% prediction accuracy for parameters with more than 2 values.

6. CONCLUSIONS AND FUTURE WORK

This paper makes several significant contributions towards creating more effective agents to interact with people in optimization and negotiation problems. First, we found that “traditional” rationality models were often poor indication how people will act. Consistent with previous research [1, 11], we found that a better alternative is to use traditional learning techniques to predict how people will behave. However, in contrast to previous works, we used decision trees [13] to formulate exactly which policy was used instead. This classifier found that bounded rationality theories, and specifically AAT, were used. Second, this paper represents a unique approach where this general theory was then reapplied to improve learning models. Within the complex negotiation domain, this approach significantly improved prediction accuracy, often by over 10%. Within the simpler optimization problem, this approach was useful in producing accurate learning models even when extremely limited learning data was available.

For future work, several directions are possible. First,

	Salary-3	Title-4	Car-2	Pension-3	Promotion-2	Hours-3	Average
Majority Rule	60.1852	67.5926	57.4074	70.3704	62.963	62.963	63.5803
Equilibrium Strategy 1	44.4444	67.5926	69.4444	66.6667	41.6667	67.5926	59.568
Equilibrium Strategy 2	25.9259	17.5926	69.4444	19.4444	43.5185	61.1111	39.5062
C4.5 Without AAT	61.111	68.5185	68.5185	67.5926	83.3333	69.4444	69.7531
C4.5 with AAT stats	62.963	68.5185	75.9259	71.2963	91.6667	76.8519	74.53705
C4.5 + Complete Knowledge	95.3704	89.814	100	96.2963	100	96.2963	96.2962

Table 1: Comparing the Prediction Accuracy between AAT and non-AAT Based Models in the Employer / Employer Negotiation Domain

	Fund Size-3	Aid-3	Zim. Tariff-2	Eng. Tariff-2	Forum-3	Average
Majority Rule	57.3333	45.3333	58.2222	62.6667	44	53.5111
Equilibrium Strategy 1	35.088	42.3849	41.962	61.9443	46.5569	45.58722
Equilibrium Strategy 2	35.088	42.3849	51.337	44.7568	41.9177	43.09688
C4.5 Without AAT	61.8257	46.888	57.7778	64	49.3776	55.97382
C4.5 with AAT stats	71.1111	56.4444	67.1111	64	55.5556	62.84444
C4.5 + Complete Knowledge	95.5556	93.7778	100	100	87.1111	95.2889

Table 2: Comparing the Prediction Accuracy between AAT and non-AAT Based Models in the Tobacco Trade Negotiation Domain

once we have demonstrated that knowing people’s aspirations improves learning, one may wish to study how these values can be quickly and accurately identified to further aid in the learning process. Second, this paper focuses on how to best learn how people interact with each other. One important application of this work is to create automated agents that use this paper’s lessons to better interact with people. State of the art automated negotiation agents [5, 6] currently do not use this information. Third, we were successful in demonstrating that AAT is more useful than traditional rationality or fast and frugal bounded heuristics to predict how people in the problems we studied. However, an open question is what other, likely bounded, general theories will be helpful in creating learning agents in other real-world environments where AAT might not be relevant.

7. REFERENCES

- [1] Y. Gal and A. Pfeffer. Predicting people’s bidding behavior in negotiation. In *AAMAS ’06*, pages 370–376, 2006.
- [2] G. Gigerenzer and D. G. Goldstein. Reasoning the fast and frugal way: models of bounded rationality. *Psychology Rev*, 103(4):650–669, 1996.
- [3] D. Kahneman and A. Tversky. Prospect theory: An analysis of decision under risk. *Econometrica*, 47:263–291, 1979.
- [4] S. Kraus, P. Hoz-Weiss, J. Wilkenfeld, D. R. Andersen, and A. Pate. Resolving crises through automated bilateral negotiations. *Artificial Intelligence*, 172(1):1–18, 2008.
- [5] R. Lin and S. Kraus. Automated negotiations: Can automated agents proficiently negotiate with humans? 53(1):78–88, 2010.
- [6] R. Lin, S. Kraus, J. Wilkenfeld, and J. Barry. Negotiating with bounded rational agents in environments with incomplete information using an automated agent. *Artificial Intelligence*, 172(6-7):823–851, 2008.
- [7] P. Maes. Artificial life meets entertainment: lifelike autonomous agents. *Commun. ACM*, 38(11):108–114, 1995.
- [8] E. Manisterski, R. Lin, and S. Kraus. Understanding how people design trading agents over time. In *AAMAS ’08*, pages 1593–1596, 2008.
- [9] Y. Murakami, Y. Sugimoto, and T. Ishida. Modeling human behavior for virtual training systems. In *AAAI*, pages 127–132, 2005.
- [10] J. V. Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1944.
- [11] D. Pardoe and P. Stone. Bidding for customer orders in tac scm. In *AAMAS 2004 Workshop on Agent Mediated Electronic Commerce VI*, pages 143–157, 2004.
- [12] D. V. Pynadath and M. Tambe. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *JAIR*, 16:389–423, 2002.
- [13] J. R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1 edition, January 1993.
- [14] A. Rosenfeld and S. Kraus. Modeling agents through bounded rationality theories. In *IJCAI-09*, pages 264–271, 2009.
- [15] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.
- [16] R. Selten. Aspiration adaptation theory. *Journal of Mathematical Psychology*, 42:1910–214, 1998.
- [17] H. A. Simon. *Models of Man*. John Wiley & Sons, New York, 1957.
- [18] E. M. Thiessen, D. P. Loucks, and J. R. Stedinger. Computer-assisted negotiations of water resources conflicts. *GDN*, 7(2):109–129, 1998.
- [19] M. L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–654, 1979.
- [20] J. Wilkenfeld, K. Young, D. Quinn, and V. Asal. *Mediating International Crises*. Routledge, London, 2005.
- [21] I. H. Witten and E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*. Morgan Kaufmann, June 2005.

Less Is More: Restructuring Decisions to Improve Agent Search

David Sarne
Department of Computer
Science
Bar Ilan University
Ramat Gan 52900, Israel

Avshalom Elmalech
Department of Computer
Science
Bar Ilan University
Ramat Gan 52900, Israel

Barbara J. Grosz
School of Engineering and
Applied Sciences
Harvard University,
Cambridge MA 02138 USA

Moti Geva
Department of Computer
Science
Bar Ilan University
Ramat Gan 52900, Israel

ABSTRACT

In many settings and for various reasons, people fail to make optimal decisions. These factors also influence the agents people design to act on their behalf in such virtual environments as eCommerce and distributed operating systems, so that the agents also act sub-optimally despite their greater computational capabilities. In some decision-making situations it is theoretically possible to supply the optimal strategy to people or their agents, but this optimal strategy may be non-intuitive, and providing a convincing explanation of optimality may be complex. This paper explores an alternative approach to improving the performance of a decision-maker in such settings: the data on choices is manipulated to guide searchers to a strategy that is closer to optimal. This approach was tested for sequential search, which is a classical sequential decision-making problem with broad areas of applicability (e.g., product search, partnership search). The paper introduces three heuristics for manipulating choices, including one for settings in which repeated interaction or access to a decision-maker's past history is available. The heuristics were evaluated on a large population of computer agents, each of which embodies a search strategy programmed by a different person. Extensive tests on thousands of search settings demonstrate the promise of the problem-restructuring approach: despite a minor degradation in performance for a small portion of the population, the overall and average individual performance improve substantially. The heuristic that adapts based on a decision-maker's history achieved the best results.

Categories and Subject Descriptors

[Agent theories, Models and Architectures]: Bounded rationality

General Terms

Human Factors, Experimentation

Keywords

Restructuring Decision Making

1. INTRODUCTION

For a variety of decision-making situations, it has been shown that people do not choose optimally or follow an optimal strategy. Research in psychology and behavioral economics has revealed various sources of this suboptimal behavior, rooted in various characteristics of human cognition and decision-making [2]. The phenomena recurs also in agents that are designed by non-specialists in decision-making theory [3]. A number of approaches have been

Cite as: Less Is More: Restructuring Decisions to Improve Agent Search, David Sarne, Avshalom Elmalech, Barbara J. Grosz and Moti Geva, *Proc. of 10th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, Tumer, Yolum, Sonenberg and Stone (eds.), May, 2–6, 2011, Taipei, Taiwan, pp. 431-438.

Copyright © 2011, International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

pursued to design computer systems which will improve the decisions made by people [13, 21]. These systems may be characterized as attempting to improve a decision-maker's judgment of a situation and her ability to identify, reason about, and compare the (full set of possible) outcomes of the different choices in a decision setting in ways that yield a better decision-making process.

In this paper we use a different approach, one that parallels recent developments in psychology and behavioral economics [18]. Instead of attempting to change a decision-maker's strategy directly so that it aligns better with the optimal one, we restructure the decision-making problem itself. This strategy mimics approaches to human decision-making that remove options to allow people to focus on an appropriate set of choices and characteristics of those choices. For instance, the buyer of a used car may reach a better decision more quickly if presented with a smaller set of possible cars and only the most important characteristics of those cars. The work described in the paper is an exploration of the hypothesis that a computer agent can make better decisions in certain settings given fewer options, with the characteristics of each option adjusted to compensate for possible reasoning biases of the agent.

Restructuring processes manipulate the choices originally available to the decision-maker. Manipulations include elimination of a subset of the alternatives available and changing the values of their characteristics. The decision-maker then makes the choices it believes to be optimal given the restructured problem. To maintain the reliability of the restructured decision setting, all choices for the manipulated problem must be legitimate choices in the original problem. Similarly, all possible outcomes of each choice in the original problem should be valid in the manipulated problem.

The advantage of the restructuring approach is that it completely avoids the need to persuade the decision-maker (either an agent or a person) of the optimality and correctness of the optimal strategy. When the space of possible strategies is complex to express or the optimal strategy is non-intuitive, substantial effort may be required for such persuasion. Restructuring is also ideal when the strategy of a searcher is pre-set and cannot be changed externally, as in the case of an autonomous agent in eCommerce. This new approach does not, however, guarantee optimality. In some cases, a few decision-makers may perform slightly less well because they lose alternatives or receive inaccurate information about the possible outcomes of the different options. The results given in this paper show that, nonetheless, overall the method substantially improves the expected outcome. While the idea that manipulating information people receive can be beneficial is not new, prior work [9, 7] mainly considered settings in which non-optimal selections derive from people's computational limitations. This paper, in contrast, deals with sequential decision-making in complex settings which require structured decision-making strategies.

The application domain used in this paper for investigating the usefulness of the problem-restructuring approach is economic search. In economic search [19], the searcher chooses one of several opportunities, each associated with a distribution of gains. The actual gain from a particular opportunity can be obtained, but there

Application	Goal	Opportunity	Value	Search Cost	Source of uncertainty
Marriage market	Maximize lifetime happiness	Date	Lifetime utility	Time spent, being alone while searching	Uncertainty regarding the potential spouse character
Job market	Optimize lifetime assets	Job interview	Offered salary and perks	Time spent, unemployment while searching	Uncertainty about potential employers
Product purchase	Minimize overall expense	Store	Product price	Time, communication and transportation expenses	Uncertainty regarding asked prices

Table 1: Mapping applications to sequential search problem

is a cost for getting it. The searcher thus needs to take into consideration the trade-off between the cost of further search and the additional benefits of it [1, 5, 11]. The sequential decision setting of economic search models a variety of daily activities. Prior literature shows that neither people nor agents they design for this problem use the optimal search strategy [15, 3]. Furthermore, the optimal solution for an economic search problem is conceptually challenging and has many inherent counter-intuitive characteristics [19] that make it difficult to persuade people to adopt it. Economic search is thus an ideal domain in which to investigate the benefits of problem-restructuring.

The paper presents three problem manipulation heuristics for the sequential search problem, two of them non-adaptive and the third adaptive. The non-adaptive heuristics apply a fixed set of manipulation rules and do not require any prior knowledge about the searcher. The first non-adaptive heuristic, denoted “information hiding”, eliminates some of the alternatives available based on the likelihood that these alternatives will not actually be needed by the optimal strategy. The second non-adaptive heuristic, denoted “mean manipulation”, attempts to manipulate the distribution of gains associated with each alternative in a way that a searcher who is influenced only by means (rather than the distribution of gains) will actually end up following the optimal strategy. Finally, an adaptive manipulation heuristic, denoted “adaptive learner”, is introduced for cases where results of prior interactions with the user are available. This heuristic attempts to model the decision-maker’s strategy and classify it according to a set of pre-defined strategies. Based on this classification, it then applies one of the non-adaptive manipulation heuristics.

We evaluate the usefulness of the problem-restructuring method and the effectiveness of the heuristics using computer agents that were programmed by students for a search domain called “job - assignment”. The results of the evaluation show that the use of manipulated choices for search problems results in search strategies that more closely resemble the optimal ones for the corresponding non-revised settings. In addition, the evaluation reveals that the heuristics differ in the nature of the improvement that individual agents achieve. With the “mean manipulation” heuristic some searchers get maximum improvement, but others substantially worsen their performance. The “information hiding” heuristic, in contrast, does not achieve the maximum possible individual improvement for any agent, but the average improvement is greater and the maximum degradation in any searcher’s expected performance is substantially smaller. As expected, the “adaptive learner” heuristic produces the best results in comparison to the non-adaptive heuristics alone.

In the following section we formally present the economic search problem, its optimal solution and the complexities associated with recognizing its optimality. Section 3 explains and justifies the agent-based methodology for testing. The three heuristics are described in Section 4, and the details of the principles used for evaluating them are given in Section 5. Section 6 summarizes the results. Section 7 surveys literature from several research fields relevant to this research. Finally we conclude in Section 8.

2. THE SEARCH MODEL

As the underlying framework for the research, we consider the canonical sequential search problem described by Weitzman [19] to which a broad class of search problems can be mapped. In this

Project	α	ω
Cost	15	20
Rewards	(0.5,100) , (0.5,55)	(0.2,240) , (0.8,0)

Table 2: Information for Simplified Example.

problem, a searcher is given a number of possible available opportunities $B = \{B_1, \dots, B_n\}$ (e.g., to buy a product) out of which she can choose only one. The value v_i to the searcher of each opportunity B_i (e.g., expense, reward, utility) is unknown. Only its probability distribution function, denoted $f_i(v)$, is known to the searcher. The true value v_i of opportunity B_i can be obtained but only by paying a fee, denoted c_i , possibly different for each opportunity. Once the searcher decides to terminate her search (or once she has uncovered the value of all opportunities) she chooses from the opportunities whose values were obtained, the one with the minimum or maximum value (depending on whether values represent costs or benefits). A strategy s is thus a mapping of a world state $W = (q, B' \subset B)$ to an opportunity $B_i \in B'$, the value of which should be obtained next, where q is the best (either maximum or minimum) value obtained by the searcher so far and B' is the set of opportunities with values still unknown. ($B_i = \emptyset$ if the search is to be terminated at this point.) The optimal sequential search strategy s^* is the one that maximizes/minimizes the expected sum of the costs incurred in the search and the value of the opportunity chosen when the process terminates.

The search problem as so formulated applies to a variety of real-world search situations. For example, consider the case of looking for a used car. Ads posted by prospective sellers may reveal little and leave the buyer with only a general sense of the true value and qualities of the car. The actual value of the car may be obtained only through a test drive or an inspection, but these incur a cost (possibly varying according to the car make, model, location, and such). The goal of the searcher is not necessarily to end up with the most highly valued car, since finding that one car may incur substantial overall cost (e.g., inspecting all cars). Instead, most car buyers will consider the tradeoff between the costs associated with further search and the marginal benefit of a better-valued opportunity. Table 1 provides mappings of other common search applications to the model. As it suggests, a large portion of our daily routine may be seen as executing costly search processes.

While the problem is common, the nature of its optimal solution is non-intuitive. A simplified version of an example from Weitzman [19] may be used to illustrate. It deals with two possible investments. The benefits of each are uncertain and can only be known if a preliminary analysis is conducted. If funds are limited, then no more than one investment would actually be carried out. Table 2 summarizes the relevant information for decision-making: investment α might yield a total benefit of 100 with probability .5 and of 55 with probability .5 and alternative investment ω with a probability of .2 might deliver a possible benefit of 240 and no benefit with probability .8. Preliminary analysis shows costs of 15 for the α investment and 20 for ω .

The problem is to find a sequential search strategy which maximizes expected value. When reasoning about which alternative to explore first, one may notice that by any of the standard economic criteria, α dominates ω . Investment α has a lower cost, higher expected reward, greater minimum reward and less variance. Consequently, most people would guess that α should be researched first [19]. However, and somewhat paradoxically, it turns out that the

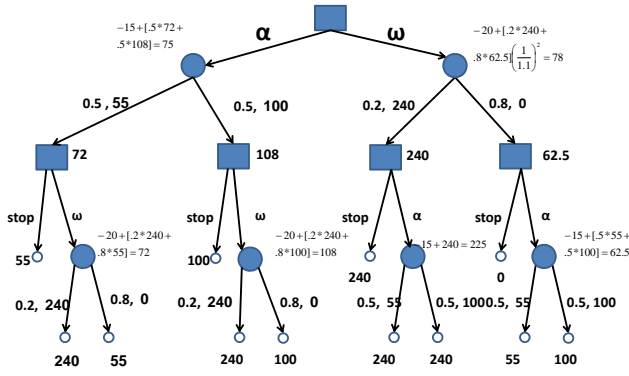


Figure 1: Solution to simplified example

optimal sequential strategy is to check ω first and if its payoff turns out to be zero then to develop α . This is shown in the decision tree in Figure 1.

It is quite simple to compute the optimal solution to the sequential search problem [19]. The solution is based on setting a reservation value (a threshold) denoted r_i for each opportunity B_i . For the expected cost minimization version of the problem, the reservation value to be used should satisfy Equation 1(a):

$$(a) \quad c_{B_i} = \int_{x=-\infty}^{r_i} (r_i - x) f_i(x) dx \quad ; \quad (b) \quad c_{B_i} = \int_{x=r_i}^{\infty} (x - r_i) f_i(x) dx. \quad (1)$$

Intuitively, r_i is the value where the searcher is precisely indifferent: the expected marginal benefit from obtaining the value of the opportunity exactly equals the cost of obtaining that additional value. The searcher should always choose to obtain the value of the opportunity associated with the minimum reservation value and terminate the search once the minimum value obtained so far is less than the minimum reservation value of any of the remaining opportunities. For revenue maximization, the reservation value should satisfy equation 1(b) and values should be obtained according to descending reservation values, until a value greater than any of the reservation values of the remaining opportunities is found.

One important and non-intuitive property of the above solution is that the reservation value calculated for each opportunity does not depend on the number and properties of the other opportunities, but rather on the distribution of the value of the specific opportunity and the cost of evaluating it.

Sequential search problems provide a good, and important, arena for investigating whether restructuring the problem is preferred over supplying the optimal search strategy to the decision-maker. As evidenced in the results section (and in prior literature [3, 15]), both people and the agents they program fail to follow the optimal strategy when engaged in sequential search. Supplying the optimal solution to the searcher may require extensive argumentation and effort because of the counter-intuitive nature of the optimal solution, in particular its myopic nature and the fact that it often favors risky opportunities [19]. A possible way to persuade a person that this is the optimal strategy is by giving her the optimality proof, but that is relatively complex and requires strong mathematical and search theory background. A possible way to persuade an agent that this is the optimal strategy is by calculating the expected value of every other sequence of decisions and compare with the expected outcome of the optimal strategy. However, the number of possible sequences for which the expected outcome needs to be calculated is theoretically infinite in the case of continuous value distributions or exponential (combinatorial) for discrete probability distribution functions. Thus, both these methods for proving optimality have substantial overhead. In contrast, the problem restructuring approach can improve performance without requiring such complex persuasion.

3. AGENT-BASED METHODOLOGY

We used computer agents rather than people to test the effectiveness of the general approach of restructuring the problem space and of the heuristics for manipulating the choices presented for several reasons. First, from a methodological perspective, this approach enables the evaluation to be carried out over thousands of different search problems, substantially improving the statistical quality of the result. Even more importantly, it eliminates people's computational and memory limitations as possible causes of inefficiency. The inefficiency of the agents' search is fully attributable to their designs, and result from the agent designers' limited knowledge of how to reason effectively in search-based environments.

Second, from an applications perspective, the ability to improve the performance of agents for search-related applications and tasks, especially in eCommerce, could significantly affect future markets. The importance and role of such agents have been growing rapidly. Many search tasks are delegated to agents that are designed and controlled by their users (e.g., comparison shopping). Many of these agents use non-optimal strategies. Once programmed, their search strategy cannot be changed externally, but it can be influenced by restructuring of the search problem.

Finally, the results from agent-based evaluations may be useful in predicting the way the proposed heuristics would affect people's search. Some prior research has shown close similarities between a computer agent's decisions and the decisions made by people in similar settings [15], in particular in search-based settings [3].

4. HEURISTICS

In this section, we define the three problem-reconstruction heuristics used in our investigations: Information Hiding, Mean Manipulation and Adaptive Learner. These heuristics differ primarily in whether they adapt to a searcher's strategy. The first two heuristics do not adapt; they assume no prior information about the searcher is available and apply a fixed set of manipulation rules. The third heuristic uses information from a searcher's prior searches to classify it and decide which of the other two heuristics to use.

For a manipulation heuristic to be considered successful, it needs not only to improve average overall agent performance, but also to avoid significantly harming the performance of any of the agents.

4.1 The Information Hiding Heuristic

This heuristic removes from the search problem opportunities for which the probability that their value will need to be obtained according to the optimal strategy s^* is less than a pre-set threshold α . By removing these opportunities, we prevent the searcher from choosing them early in the search, yielding a search strategy that is better aligned with the optimal strategy in the early, and more influential, stages of the search. While the removal of alternatives is likely to worsen the performance of fully rational agents (ones that use the optimal strategy), the expected performance decrease is small; the use of the threshold guarantees that the probability is relatively small that these removed opportunities are actually required in the optimal search.

Formally, for each opportunity B_i we calculate its reservation value, r_i , according to Equation 1. The probability of needing to obtain the value of opportunity B_i according to the optimal strategy, denoted P_i , is given by $P_i = \prod_{r_j \leq r_i} P(v_j \geq r_i)$ for the cost minimization version of the problem, and $P_i = \prod_{r_j \leq r_i} P(v_j \leq r_i)$ for its revenue maximization version. The heuristic omits from the problem every opportunity B_i ($i \leq n$) for which $P_i \leq \alpha$.

4.2 The Mean Manipulation Heuristic

This heuristic addresses the problem that people tend to overemphasize mean values, reasoning about this one feature of a distribution rather than the distribution more fully. Their search strategies typically choose to obtain the value of the opportunity for which the difference between its expected net value and the best value obtained so far is maximal. We denote this strategy "naive mean-

based greedy search”. Formally, denoting the mean of opportunity B_i by μ_i , searchers using the naive mean-based greedy strategy calculate for each opportunity the value $w_i = \mu_i - c_i$ (in the cost maximization version) or $w_i = \mu_i + c_i$ (in the revenue minimization version) and choose to obtain the value of opportunity $B_i = \operatorname{argmax}_{B_j} \{w_j | B_j \in B' \cap w_j \geq v\}$ (or $\operatorname{argmin}_{B_j} \{w_j | B_j \in B' \cap w_j \leq v\}$ in the cost minimization version), where v is the best value obtained so far.

The heuristic restructures the problem such that w_i of each opportunity B_i in the restructured problem equals the reservation value r_i calculated for that opportunity in the original problem. This ensures that the choices made by agents that use the naive mean-based greedy search strategy for the restructured problem are fully aligned with those of the optimal strategy for the original problem. The restructuring is based on assigning to each opportunity B_i ($0 < i \leq n$) a revised probability distribution function f'_i such that $w'_i = r_i$, where r_i is the reservation value calculated according to Equation 1. The manipulation of f_i is simple, as it only requires allocating a large mass of probability around μ'_i (the desired mean, satisfying $w'_i = r_i$). The remaining probability can be distributed along the interval such that μ'_i does not change.

4.3 The Adaptive Learner Heuristic

The adaptive learner heuristic attempts to classify the strategy of a searcher and uses this classification to determine the best problem restructuring method to apply. For this purpose we need to have a representative strategy for each strategy class that has all the typical characteristics of strategies in the class. The heuristic thus requires the development of agents, denoted “class-representing agents”. Each “class-representing agent” employs the representative-strategy of its class. The measure of similarity between any searcher’s strategy and a given class is the relative distance between its performance and the performance of the class-representing agent for that class over the same set of problems. The searcher is classified as belonging to the class for which the relative performance distance to its representing-agent is minimal, and below a threshold γ . Otherwise, it is classified as belonging to a default class. Once the searcher is classified, the restructuring heuristic for its class can be applied. The use of the threshold γ assures that for any agent that cannot be accurately classified, a default manipulation heuristic is used, one that guarantees no substantial possible degradation in the performance of the agent. The adaptive learner heuristic is given in Algorithm 1.

Algorithm 1 Adaptive Learner

Input: O - Set of prior problem instances.

S - Set of strategies.

Threshold - classification threshold.

Output: s^* - the classification strategy for the searching agent (*null* if not classified or no previous data).

```

1: Initialization:  $d_s \leftarrow 0 \forall s \in S$ 
2: for every  $o \in O$  do
3:   for every  $s \in S$  do
4:      $d_s \leftarrow d_s + \frac{\|Performance_{agent}(o) - Performance_s(o)\|}{Performance_s(o)}$ 
5:   end for
6: end for
7: if  $\min\{d_s\} \leq Threshold$  then
8:   return  $\operatorname{argmin}_s(d_s)$ 
9: else
10:  return null
11: end if

```

The algorithm receives as an input the results of prior searches and a set S of strategy classes. The function $Performance_s(o)$ returns the performance of the class-representing agent $s \in S$ given the problem instance o (where $Performance_{agent}(o)$ is the perfor-

mance of the searcher being classified based on o). The algorithm returns the strategy s^* to which the agent is classified (or *null*, if none of the distance measures are below the threshold set). Based on the strategy returned we apply the manipulation heuristic which is most suitable for this strategy type (or the default manipulation if *null* is returned).

The results we report in this paper are based on two strategy classes: optimal strategy and naive mean-based greedy search strategy.¹ For a searcher that cannot be classified as one of these two strategies, we use the information hiding manipulation. To produce the functionality $Performance_s(o)$ required in Algorithm 1, we developed the following two agents:

- *Optimal Agent*. This agent follows Weitzman’s optimal solution [19].
- *Mean-based Greedy Agent*. This agent follows the naive mean-based greedy search strategy described in Subsection 4.2.

The more observations of prior searcher behavior that the adaptive heuristic’s algorithm is given, the better the classification it can produce, and consequently the better the searcher’s performance is likely to be after the appropriate manipulation is applied. Obviously, an even greater improvement in performance could be obtained if the heuristic had access to the searcher (i.e., the agent) rather than just the records describing prior searches. If direct access to the agent is allowed, then a straightforward improvement of the method would be executing the agent over each set of choices obtained, using any of the different methods and classifying it accordingly.

5. EVALUATION

To evaluate the three heuristics and our hypothesis that agents’ performance in search-based domains can be improved by restructuring the search problem, we used a search domain called “job-assignment”. Job-assignment is a classic server-assignment problem in a distributed setting that can be mapped to the general search problem discussed in this paper. The problem considers the assignment of a computational job for execution to a server chosen from among a set of homogeneous ones (servers). The servers differ in the length of their job queue. Only the distribution of each server’s queue length is known. To learn the actual queue length of a server, it must be queried, an action that takes some time (server-dependent). The job can eventually be assigned only to one of the servers that were queried. The goal is to find a querying strategy that minimizes the overall time until the job starts executing. The mapping of this problem to the sequential search problem (in its cost minimization variant) is straightforward: each server represents an opportunity where its queue length is its true value and the querying time is the cost of obtaining the value of that opportunity.

5.1 Agent Development

The evaluation used agents designed by computer science students in a core Operating Systems course. While this group does not represent human searchers in general, it fairly represents future agent developers who are likely to design the search logic for eCommerce and other computer-aided domains. As part of her regular course assignment, each student created an agent that receives as input a list of servers, their distribution of waiting times and querying costs (times); queries the servers (via a proxy program) to learn its associated waiting time; and then chooses one of them for executing a (dummy) program. The students’ grade in the assignment was correlated with their agent’s performance, i.e., the time it takes until the program is executed on one of the servers.

¹We use the optimal strategy class as a means for representing the class of strategies that are better off without applying problem restructuring. The optimal strategy is part of this class, though, as reported in the evaluation section, none of the agents we evaluated actually used the optimal strategy.

As part of their assignment, students provided documentation that described the algorithm used for managing the search for a server.

An external proxy program was used to facilitate communication with the different servers. The main functionality of the proxy was to randomly draw a server's waiting time, based on its distribution, if queried, and to calculate the overall time elapsed from the beginning of the search until the program is assigned to a server and starts executing (i.e., after waiting in the server's queue). To simplify the search problem representation, distributions were formed as multi-rectangular distribution functions. In multi-rectangular distribution functions, the interval is divided into sub intervals x_0, \dots, x_n and the probability distribution is given by $f(x) = \frac{P_i}{x_i - x_{i-1}}$ for $x_{i-1} < x < x_i$ and $f(x) = 0$ otherwise, ($\sum_{i=1}^n P_i = 1$). The benefit of using a multi-rectangular distribution function is its simplicity and modularity, in the sense that any distribution function can be modeled through it with a small number of rectangles.

5.2 Analysis Methodology

The agents that the students developed were executed on a set of problems with the full set of search choices and no restructuring. The problems in the set varied in their characteristics (e.g., number of opportunities, characteristic of distribution functions, querying costs). Each agent was then run on each problem restructured according to the different restructuring heuristics. The performance of the agents was logged. In parallel, the class-representing optimal and mean-based greedy agents were executed over the same problem set. The results obtained by the students' agents on the non-manipulated problem set were compared with their results on the restructured problems. The results of the optimal agent were used as a baseline for evaluating the improvement achieved by each of the restructuring heuristics. Results were tested for statistical significance using t-test (with $\alpha = 0.05$), whenever applicable.

The designs of the students' agents were also analyzed to identify a set of common search strategy characteristics. We then looked for common features among agents that performed similarly.

5.3 Performance Measures

The evaluation of the different heuristics used two complementary measures: (1) relative decrease in the time until the job is executed; and (2) the relative reduction in search inefficiency. Formally, we denote the expected time until execution for the optimal search strategy by t_{opt} and the time until execution for an agent on the manipulated and non-manipulated problem by t_{man} and t_{-man} , respectively. The first measure, calculated as $\frac{t_{-man} - t_{man}}{t_{-man}}$, relates directly to the time saved. It depends on the problem set, because t_{-man} can vary widely. The second measure, calculated as $\frac{t_{man} - t_{opt}}{t_{-man} - t_{opt}}$, takes into account that the search time using either the manipulated or original data is bounded by the performance of the optimal agent. It thus highlights the efficiency of the heuristic in improving performance.

For each of the two measures, the average over the entire set of problem instances and across all agents was calculated from both social and individual perspectives. For the social perspective, we calculated the relative improvement in both measures over the aggregated times obtained for all agents in all problems. For the individual perspective, we calculated the average of individual improvements for both measures. For each evaluated heuristic the maximum decrease in individual average performance was also identified, because an important requirement for a successful heuristic is that it does not substantially worsen any of the agents' individual performance.

6. RESULTS AND ANALYSIS

Seventy six agents, each designed by a different student, were used to evaluate the heuristics. The test set used for evaluation consisted of 5000 problems that were generated with a random number of servers in the range (2,20), costs of querying the differ-

ent servers uniformly drawn from the range (1,100), and a multi-rectangular distribution function to each server generated by randomly setting a width and probability for each rectangle and then normalizing it to the interval (0,1000).

In this section we present the main analysis carried out over these agents using this problem set. The results using two other problem sets are given in Subsection 6.5.

6.1 Agent Strategy

The strategies students used reveal several characteristics along which agent designs vary when programmers who are not search experts do the design. Our analysis of the agents using program documentation (and occasionally the code itself) revealed several problem features commonly used in their search strategies, including expected value (in 41 of the agents), variance (in 6 of the agents), and the median (in 2 of the agents) of each server. Additional factors used in some designs were the time cost of querying servers (in 37 of the agents), randomness in the decision-making process (in 11 of the agents), a preliminary selection of servers for querying (in 57 of the agents), the inclusion of the cost incurred so far (i.e., "sunk cost") in the decision-making process (in 4 of the agents) and the use of the probability of finding a server with a lower waiting time than the minimum found so far (in two of the agents).

Several interesting observations may be made based on these characteristics. First, many of the agents use the mean waiting time of a server as a parameter that directly influences the search strategy, even though the optimal strategy is not affected directly by means (see Section 2). Second, a substantial number of agents (39 of 76) do not take into account the cost of search in their strategy. One possible explanation for this phenomena is that the designers of these strategies considered cost to be of very little importance in comparison to the mean waiting times. Interestingly, several students (11 of 76) use randomization in their search strategy, even though, as explained in Section 2, randomness is not useful for these problems (and plays no role in the optimal strategy).

The average performance of the different agents on the 5000 test problem instances is given in Figure 2. The vertical axis represents the average overall time until execution and the horizontal axis is the agent id. The two horizontal lines in the figure represent the performance of an agent searching according to the optimal strategy and an agent using the random selection rule, "randomly query a random number of servers and assign the job to the server with the lowest waiting time". As can be seen from the figure, none of the students' agent strategies reached the performance of the optimal strategy. The average overall time obtained by the agents is 445.77, while that of the optimal agent is 223.1. Furthermore, many of the strategies (41 out of 76) did even worse than a random agent.

We attempted to identify clusters of agents, based on agent performance and characteristics of agent design, as identified by our analysis of agent designs. The following clusters emerged from this assessment:

- The naive mean-based greedy search strategy and its variants (e.g., agents 3-7).
- Mean-based approaches that involve preliminary filtering of servers according to means and costs (e.g., agents 15-17).
- A variation of the naive mean-based greedy search strategy that also takes the variance of each server as a factor (e.g., agents 22-23).
- Querying the two servers with the lowest expected queue length and assigning the job to the one with the minimum value found (e.g., agents 24-27).
- Assigning the job to the first/last/random server (e.g., agents 42-71).

For many agents, similarities in performance could not be explained by resemblances among the strategies themselves. Although in some cases, agents used different variants of the same basic strategy, apparently the differences among the variants resulted in substantial differences in performance. The most interesting and

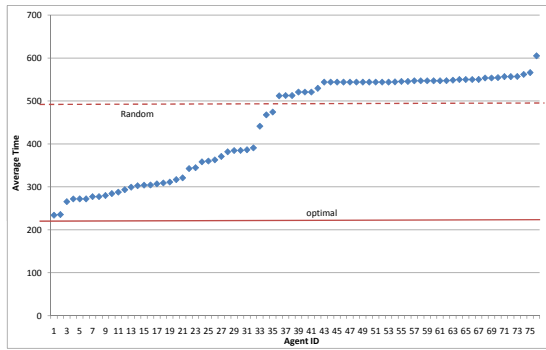


Figure 2: Agent performance without any manipulation (to make the results easier to follow, IDs are ordered based on performance)

unique strategies deployed include: (a) adding up the costs of the servers already queried, and based on this sum deciding whether to continue to the next server or to assign the job to the server with the lower execution time found so far; (b) taking 10% of the servers with the highest variance and querying them one by one, until the real value of one of them is less than the one of the server with the minimal expected value.

There was one major distinction between agent designs that led us to separate out a group of agents. Although many of the agents used search strategies that took into account affected information obtained in searching, a significant number did not follow any sequential decision-making rule, but rather queried only one server chosen arbitrarily. While any selection rule was considered legitimate for the students’ assignment, strategies of the latter type are not true search strategies. Because these strategies are simple for the adaptive learner to identify (they choose a single server according to a simple pattern) and a simple problem reconstruction method could easily improve their behavior (provide only one choice: the server with the minimum sum of expected waiting time and querying time), we removed the results for these agents (agents 32-75) from the main analyses given in this paper. If included in the analysis, the improvement in the average overall performance of the adaptive agent reported in the following subsections would have been substantially better.

6.2 Analysis of Information Hiding

The threshold α used for removing alternatives from the problem instance is a key parameter affecting the “Information Hiding” heuristic. Figure 3 depicts the average time until execution (over all agents, for the 5000 problem instances) for different threshold values. For comparison purposes, it also shows the average performance on the non-manipulated set of problems, which corresponds to $\alpha = 0$ (the horizontal line). The shape of the curve has an intuitive explanation. First, for small threshold values, an increase in the threshold increases agent performance as it further reduces the possible deviation from the optimal sequence. However, as the threshold increases, the probability increases that the opportunities this increase allows to be removed are ones that would be examined by the optimal strategy.

As the graph in Figure 3 shows, the optimal threshold is $\alpha = 10\%$, for which an average time of 299.33 is obtained. This graph also shows that for a large interval of threshold values around this point — in particular for $3\% \leq \alpha \leq 30\%$ — the performance level is similar. Thus, the improvements are not extremely sensitive to the exact value; relatively good performance may be achieved even if the α value used is not exactly the one which yields the minimum average time. In fact, any threshold below $\alpha = 55\%$ results in improved performance in comparison to the performance obtained

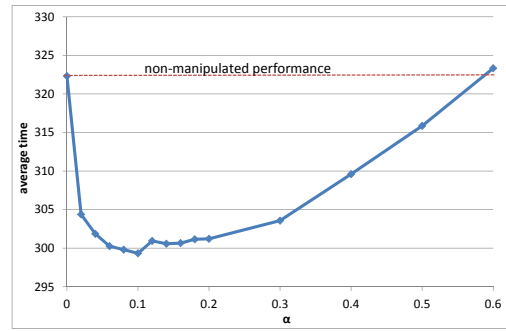


Figure 3: The effect of α in “information hiding” over average performance (cross-agents)

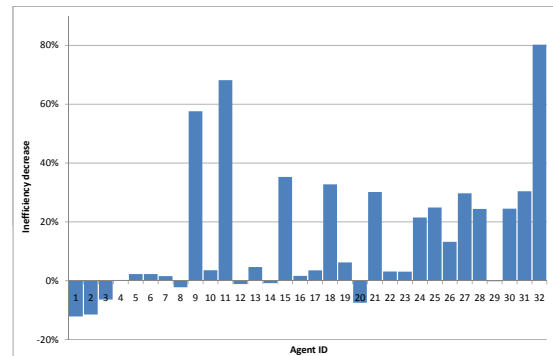


Figure 4: Average reduction in the search inefficiency of information hiding for $\alpha = 10\%$

without the use of this manipulation heuristic (i.e., with $\alpha = 0$).

Figure 4 depicts the average reduction in search inefficiency (over the 5000 problem instances) of each agent with $\alpha = 10\%$. As the figure shows, this heuristic decreased the search inefficiency of 24 of the 32 agents. The maximum improvement was obtained by agent 32 (80.18%). The average reduction (individual welfare) is 14.49% and the overall reduction (social welfare) is 5.52%. The downside of this heuristic is that it increases the overhead of some of the agents’ searches. The highest increase in the overhead of any agent was, however, minimal and equals 12.1% (for agent 1), corresponding to an increase of 0.57% in its average time until its job starts executing.

The main advantages of this strategy are that it improves the performance of most agents and that even in cases in which an individual agent’s performance degrades, the degradation is relatively small. Thus, the heuristic is a good candidate for use as a default problem-restructuring heuristic whenever there is no information about searcher strategy or an agent cannot be classified accurately.

6.3 Analysis of Mean Manipulation

Figure 5 depicts the average reduction in search inefficiency, using the same standard problem set, of each agent when using the “Mean Manipulation” heuristic. With this heuristic, seven agents almost fully eliminate their search overhead. These agents use variants of the naive mean-based greedy search strategy. Other agents also benefited from this heuristic and substantially reduced the overhead associated with their inefficient search. These agents (e.g., 4, 7, 25) all also included mean-based considerations, to some extent, in their search strategy.

This heuristic has a significant downside, however. Ten agents did worse with the mean manipulation heuristic, 5 of them substantially worse. The search overhead of these agents, in comparison to optimal search, increased by 50-250%. With this heuristic the overall inefficiency (social welfare) actually increased by 2.5%

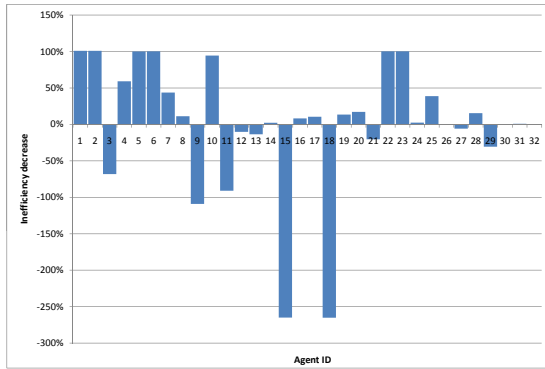


Figure 5: Average reduction in the search inefficiency of Mean Manipulation

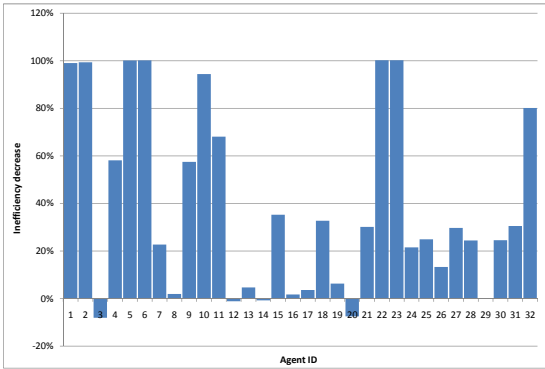


Figure 6: Average reduction in the search inefficiency of Adaptive Learner

even though the average overhead decreased by 1.3%, a classical case of Simpson’s paradox [17]. The substantial increase in search overhead for some of the agents makes this heuristic inappropriate for general use. It is, however, very useful when incorporated into an adaptive mechanism that attempts to identify those agents that use mean-based strategies and applies this manipulation method on their input.

6.4 Analysis of Adaptive Learner

The adapter learner has the best of both worlds. Figure 6 depicts the average reduction in search inefficiency, using the standard problem set, of each agent when using the adaptive learner heuristic with $\gamma = 10\%$. For agents that were badly affected by the mean-manipulation heuristic, the information hiding manipulation was used instead, improving their performance. Overall, the inefficiency (social welfare) decreased by 37.4%. The average reduction in individual inefficiency (individual welfare) is 39%. Out of the 32 agents, 5 agents slightly worsened their performance (maximum of 8% increase in inefficiency, which is equivalent to a 1.3% increase in the expected waiting time of that agent).

6.5 Evaluation with Different Problem Sets

To show that the results were not due to a wise selection of problem instance characteristics, we repeated the evaluation with other distributions of queue lengths and different querying costs. Two problem sets were used,

- Increased possible querying time (denoted “Inc Quer”): same as the original set of problems, except that the querying time was taken from an interval that was three times as large (resulting in an increased ratio between querying time and possible waiting times in queue).
- Increased queue time variance (denoted “Inc Var”): same as the original set of problems, except that the possible waiting

time interval was increased from 1,000 to 10,000 (resulting in a substantial increased variance in server waiting time in queue).

Each of these problem sets also contains 5000 different problems.

Table 3 presents the results obtained for the new problem sets in comparison to the original set. As can be seen from the table, the improvement obtained from the different heuristics is consistent with the one obtained using the original set.

	Original	Inc. Var	Inc. Quer.
Non-Manipulated	322.3	2449.4	461.9
Adaptive	223.1	2203.0	388.1
Optimal	285.2	1349.7	332.3
Average individual improvement	10.2%	9.7%	10.5%
Overall (social) improvement	11.5%	10.1%	16.0%
Maximum individual performance decrease	2.2%	2.0%	1.9%
Average individual inefficiency reduction	39.0%	29.0%	46.5%
Overall inefficiency reduction	37.4%	22.4%	56.9%

Table 3: Performance for different classes of problems

7. RELATED WORK

People are bounded rational [16], unlike computer agents, which may deploy rational strategies and are significantly less bounded computationally. They cannot be trusted to exhibit optimal behavior [14]. Furthermore, people often tend not to use the optimal strategy even when one is provided [10]. Their decision-making may be influenced by selective search, the tendency to gather facts that support certain conclusions while disregarding other facts that support other conclusions [2], and by selective perception — the screening-out of information that one does not think is important [6]. Others [18], have attributed people’s difficulty in decision-making to the conflict between a “reflective system” (e.g., involved in decisions about which college to attend, where to go on trips and in most circumstances, whether or not to get married) and an “automatic system” (e.g., that leads to smiling upon seeing a puppy, getting nervous while experiencing air turbulence, and ducking when a ball is thrown at you).

Over the years a variety of work has addressed the challenge of improving people’s decision-making, mostly by developing decision support systems to assist users in gathering, merging, analyzing, and using information to assess risks and make recommendations in situations that may require tremendous amounts of the users’ time and attention [21]. Recently, several approaches have been proposed that attempt to reconstruct the decision-making problem [18] instead of attempting to change people’s decision-making strategies directly. This prior work focused on psychological aspects of human decision-making, and does not involve any learning or adaptation. Furthermore, none of this prior work dealt with a sequential decision-making process.

The search model discussed in this paper, which considers an optimal stopping rule for individuals engaged in costly search (i.e., ones for which there is a search cost) builds on economic search theory,² and in particular its sequential search model [12]. While search theory is a rich research field, its focus is on the theoretical aspects of the optimal search strategy and it does not address the non-optimality of search strategies used by people or rationally-bounded agents.

A range of research in multi-agent systems has examined people’s use of agents designed to represent them and act on their behalf. For example, Kasba [4] is a virtual marketplace on the Web

²A literature review of search theory may be found elsewhere [12].

where people create autonomous agents in order to buy and sell goods on their behalf. Various research has involved programming agents in the decision-theoretic framework of the Colored-Trails game [8]. Here, the agents had to reason about other agents' personalities in environments in which agents are uncertain about each other's resources. In the Trading Agent Competition (TAC) [20], agents are used to collect people's strategies. Work involving people who design agents provides some evidence that people fail to build in the optimal strategy [3], in particular in search-based environments [15]. This work has not, however, provided methods for improving the performance of such agents through problem restructuring of any sort.

8. DISCUSSION AND CONCLUSIONS

The results reported in Section 6 are encouraging and a proof of concept for the possibility of substantially improving agent performance in sequential search by restructuring the problem space. The extensive evaluation reveals that even with no prior information regarding an agent's strategy, a heuristic such as information hiding produces substantial improvement in average performance while limiting individual potential performance degradation. With even limited information about the prior search behavior of an agent, heuristics such as the adaptive learner can further improve the overall performance and lower even further the possible decrease in individual agent performance. These results were consistent across three different classes of search environments in extensive evaluations involving a large number of agents, each designed by a different person, and a large number of problems within each class.

Restructuring of the problem space is applicable in settings for which the optimal choice cannot be revealed but rather an optimal sequential exploration should be devised, and the optimal exploration strategy cannot be provided directly to the decision-maker or the decision-maker cannot easily be convinced of its optimality. Instead, we can only control the information the decision-maker obtains in the problem.

The problem restructuring technique has great potential for market designers (who also have the domain-specific information that can lead to more intelligent restructuring heuristics). Consider for example large scale Internet websites like *autotrader.com* or *expedia.com*. These web-sites attempt to attract as many users as possible to increase their revenues from advertisements. Every listing for a flight or a car on these web-sites is an opportunity that needs to be explored further to realize its true value to the user. The welfare of users or the agents they use can thus be substantially improved by manipulating the listings.

The heuristic that provides the best performance is the adaptive learner. As our ability to recognize and differentiate additional strategy clusters and produce appropriate choice manipulations for them improves, we expect the performance improvement obtained by applying the adaptive strategy to increase even further. The adaptive heuristic's architecture is modular, allowing its augmentation using the new manipulation heuristics to be straightforward.

The research reported in this paper is, to the best of our knowledge, the first to attempt to restructure the decision-making problem in order to improve performance in a sequential decision-making setting. The fact that the searcher is facing a sequence of decisions and all manipulations over the choices take place prior to beginning the process, substantially increases the complexity for heuristics. In this case the search strategy used by the searcher becomes more complex as her decisions are also affected by the temporal nature of the problem and the new data that is being obtained sequentially. The challenge faced by the manipulation designer is thus substantially greater than in one-shot decision processes. In the latter case many simple and highly efficient manipulation techniques can be designed. For example, if the searcher is limited to obtaining the value of only one opportunity overall, the simplest and most efficient choice of a manipulation technique would be to remove all opportunities other than the optimal one.

A natural extension of this work involves developing heuristics

that will choose the manipulation method to be applied not only based on agent classification but also based on problem instance characteristics. This, of course, requires a more refined analysis in the agent level.

Acknowledgments

This work was supported by ISF/BSF grants 1401/09 and 2008-404.

9. REFERENCES

- [1] Y. Bakos. Reducing buyer search costs: Implications for electronic marketplaces. *Mgmt. Science*, 42:1676–92, 1997.
- [2] G. Blackhart and J. Kline. Individual differences in anterior EEG asymmetry between high and low defensive individuals during a rumination/distraction task. *Personality and Individual Differences*, 39(2):427–437, 2005.
- [3] M. Chalamish, D. Sarne, and S. Kraus. Programming agents as a means of capturing self-strategy. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1161–1168, 2008.
- [4] A. Chavez and P. Maes. Kasbah: An agent marketplace for buying and selling goods. In *PAAM*.
- [5] S. Choi and J. Liu. Optimal time-constrained trading strategies for autonomous agents. In *Proc. of MAMA'2000*, 2000.
- [6] A. Drake Roger. Processing persuasive arguments: Discounting of truth and relevance as a function of agreement and manipulated activation asymmetry. *Journal of Research in Personality*, 27(2):184–196, 1993.
- [7] Y. Elmaliach and G. Kaminka. Robust multi-robot formations under human supervision and control. *Journal of Physical Agents*, 2(1):31, 2008.
- [8] B. Grosz, S. Kraus, S. Talman, B. Stossel, and M. Havlin. The influence of social dependencies on decision-making: Initial investigations with a new game. In *AAMAS*, pages 780–787, 2004.
- [9] S. Iyengar. *The Art of Choosing*. Twelve, 1 edition, March 2010.
- [10] D. Kahneman and A. Tversky. *Choices, values, and frames*. Cambridge University Press, New York, 2000.
- [11] J. Kephart and A. Greenwald. Shopbot economics. *JAAMAS*, 5(3):255–287, 2002.
- [12] J. McMillan and M. Rothschild. Search. In R. Aumann and S. Hart, editors, *Handbook of Game Theory with Economic Applications*, pages 905–927. 1994.
- [13] D. Power and R. Sharda. Decision support systems. *Springer Handbook of Automation*, pages 1539–1548, 2009.
- [14] M. Rabin. Psychology and economics. *Journal of Economic Literature*, pages 11–46, March 1998.
- [15] A. Rosenfeld and S. Kraus. Modeling agents through bounded rationality theories. In *IJCAI'09*, pages 264–271, 2009.
- [16] H. Simon. Theories of bounded rationality. *Decision and organization: A volume in honor of Jacob Marschak*, pages 161–176, 1972.
- [17] E. H. Simpson. The interpretation of interaction in contingency tables. 1951.
- [18] R. Thaler and C. Sunstein. *Nudge: Improving decisions about health, wealth, and happiness*. Yale Univ Pr, 2008.
- [19] M. L. Weitzman. Optimal search for the best alternative. *Econometrica*, 47(3):641–54, May 1979.
- [20] www.sics.se/tac.
- [21] E. S. Yu. Evolving and messaging decision-making agents. In *AGENTS '01*, pages 449–456, 2001.

Author Index

- Ágotnes, Thomas, 735
- Aadithya, Karthik .V., 1121
Agmon, Noa, 91, 1103
Agogino, Adrian, 1157
Ahrndt, Sebastian, 1257
Albayrak, Sahin, 1257, 1325
Alberola, Juan M., 1221, 1349
Alcântara, João, 1275
Alcalde, Baptiste, 895
Aldewereld, Huib, 1231
Alechina, Natasha, 397
Alers, Sjriek, 1311
Almagor, Shaull, 319
Altakrori, Malek H., 1163
Amato, Christopher, 1149
Amgoud, Leila, 1237
Amigoni, Francesco, 99
An, Bo, 609, 1101
André, Elisabeth, 441, 1093
Antos, Dimitrios, 1097, 1337
Apolloni, Andrea, 693
Arcos, Josep Lluís, 669
Arellano, Diana, 1093
Argente, Estefania, 1191
Atkinson, Katie, 905
Au, Tsz-Chiu, 1225
Aylett, Ruth, 1117, 1119
Aziz, Haris, 183, 191
- Böhm, Klemens, 241, 795
Bachrach, Yoram, 1179, 1197
Bagnell, J. Andrew, 207
Bagot, Jonathan, 1319
Baier, Jorge A., 1267
Balbiani, Philippe, 1207
Baldoni, Matteo, 467
Balke, Tina, 1109
Baltes, Jacky, 1319
Baroglio, Cristina, 467
Barrett, Samuel, 567
Bartos, Karel, 1123
Basilico, Nicola, 99, 1317
Baumeister, Dorothea, 853
Bee, Nikolaus, 1093
Beetz, Michael, 107
Bentahar, Jamal, 483
Bentor, Yinon, 769
Bibu, Gideon D., 1339
Billhardt, Holger, 1243
Black, Elizabeth, 905
Bloembergen, Daan, 1105, 1311
Bošanský, Branislav, 989, 1273, 1309
- Boella, Guido, 1203
Boerkoel, James C., 141
Bonzon, Elise, 47
Botia, Juan, 1215
Botti, Vicente, 929, 1191, 1241, 1305
Boucké, Nelis, 803
Boukricha, Hana, 1135
Bowring, Emma, 133, 457
Brânzei, Simina, 1281
Brandt, Felix, 183
Brazier, Frances, 389
Broda, Krysia, 1137
Brooks, Logan, 1239
Brooks, Nathan, 1245
Brown, Matthew, 457
Bsufka, Karsten, 1325
Bulling, Nils, 275, 1187
Burguillo-Rial, Juan C., 669
Bye, Rainer, 1325
- Cai, Kai, 879
Caire, Patrice, 895
Calisi, Daniele, 1327
Caminada, Martin, 1127, 1307
Cap, Michal, 1201
Carlin, Alan, 157, 1149
Carnevale, Peter, 937
Carvalho, Arthur, 635
Cavalcante, Renato L.G., 165, 1099
Cavazza, Marc, 449, 1323
Centeno, Roberto, 1243
Ceppi, Sofia, 981, 1125
Cerquides, Jesus, 133, 379
Chaib-draa, Brahim, 947
Chakraborty, Nilanjan, 685
Chalkiadakis, Georgios, 787
Chandramohan, Mahinthan, 1321
Chang, Yu-Han, 1313
Charles, Fred, 449, 1323
Chen, Shijia, 1301
Chen, Xiaoping, 1301
Chen, Yiling, 175, 627
Chen, Yingke, 1229
Cheng, Chi Tai, 1319
Cheng, Min, 1301
Cheng, Shih-Fen, 1147
Cheng, Yong Yong, 1321
Chernova, Sonia, 617
Chhabra, Meenal, 63, 415
Chinnow, Joël, 1325
Chiou, Che-Liang, 643
Chiu, Chung-Cheng, 1023
Chopra, Amit K., 467, 475

Cigler, Ludek, 509
Cohn, Robert, 1287
Colombo Tosatto, Silvano, 1203
Comanici, Gheorghe, 1079
Conitzer, Vincent, 327, 1013
Corruble, Vincent, 701
Crandall, Jacob W., 1163, 1265
Cranefield, Stephen, 1113, 1181
Criado, Natalia, 1191, 1331
Crosby, Matt, 1213
Cullen, Shane, 37

D'Agostini, Andrea, 1327
d'Avila Garcez, Artur, 1203
Dahlem, Dominik, 601
Damer, Steven, 1255, 1367
Das, Sanmay, 63, 415
Dasgupta, Prithviraj, 1217
Dastani, Mehdi, 301, 405, 1187, 1201
Datta, Anwitaman, 1071
Dechesne, Francien, 1205
Decker, Keith S., 13
Deconinck, Geert, 803
Decraene, James, 1321
Degris, Thomas, 761
Delle Fave, Francesco M., 371
Dellunde, Pilar, 971
Delp, Michael, 761
del Val, E., 1241, 1347
Devlin, Sam, 225, 1227
Dey, Anind K., 207
De Craemer, Klaas, 803
De Hauwere, Yann-Michaël, 1115
de Jong, Steven, 551, 1311
de Keijzer, Bart, 191
de Melo, Celso M., 937
De Vos, Marina, 1109
Dias, João, 1117
Dias, M. Bernardine, 1247
Dibangoye, Jilles S., 947
Dignum, Frank, 921, 1249, 1291
Dignum, Virginia, 1205, 1231, 1291
Dikenelli, Oguz, 1335
Doherty, Patrick, 743
Dolan, John M., 753
Dolha, Mihai Emanuel, 107
Dorer, Klaus, 1199
Doshi, Prashant, 1229, 1259
Dowling, Jim, 601
Dssouli, Rachida, 483
Dudík, Miroslav, 1165
Duggan, Jim, 1211
Dunin-Kępcicz, Barbara, 743
Duong, Quang, 1351
Durfee, Edmund H., 29, 141, 1287

Dziubiński, Marcin, 1171

Eck, Adam, 1283
El-Menshawy, Mohamed, 483
Elkind, Edith, 55, 71, 821
Elmalech, Avshalom, 431
Emele, Chukwuemeka D., 913
Endrass, Birgit, 441
Endriss, Ulle, 79
Enz, Sibylle, 1119
Epstein, Leah, 525
Epstein, Shira, 457
Erdélyi, Gábor, 837
Ermon, Stefano, 1277
Erriquez, Elisabetta, 1085, 1353
Espinosa, Agustin, 1189
Esteva, Marc, 1131

Fabregues, Angela, 1315
Fagyal, Zsuzsanna, 693
Faliszewski, Piotr, 821
Faltings, Boi, 509
Fan, Xiuyi, 1095, 1341
Farinelli, A., 363
Fatima, Shaheen, 1083
Fedi, Francesco, 1327
Flacher, Fabien, 701
Fleming, Michael, 871
Fridman, Natalie, 457, 1365

Gairing, Martin, 559
Gal, Ya'akov (Kobi), 345, 551
Ganzfried, Sam, 533, 1111
García-Fornes, Ana, 929, 1189, 1221
Garrido, Antonio, 1305
Gatti, Nicola, 199, 981, 1125, 1317
Gelain, Mirco, 1209
Genovese, Valerio, 1203
Gerding, Enrico H., 811
Geva, Moti, 431
Gimeno, Juan A., 1305
Gini, Maria, 1255
Giret, Adriana, 1305
Glinton, Robin, 677
Gmytrasiewicz, Piotr, 1285
Godo, Lluís, 971
Goh, Wooi Boon, 1091
Gomes, Carla, 1277
Gomes, Paulo F., 1039
Goodrich, Michael A., 1265
Goranko, Valentin, 727
Graepel, Thore, 1179
Gratch, Jonathan, 937, 1289
Greenwood, Dominic, 1199
Grill, Martin, 1123
Grosz, Barbara J., 431

Grunewald, Dennis, 1325
 Grześ, Marek, 963, 1227
 Guiraud, Nadine, 1031, 1207
 Guo, Qing, 1285
 Guzman, Emitza, 107

 Höning, Nicolas, 1293
 Hütter, Christian, 241
 Hadad, Meirav, 1177
 Haghpanah, Yasaman, 1375
 Harbers, Maaïke, 1201
 Harland, James, 1139
 Harrison, William, 601
 Hasegawa, Takato, 1173
 Hassan, Yomna M., 1163
 Hazon, Noam, 71
 Heßler, Axel, 1257
 Hennes, Daniel, 551, 1311
 Hernández, Carlos, 123, 1267
 Herzig, Andreas, 1207
 Hindriks, Koen V., 275
 Hirsch, Benjamin, 1257
 Ho, Chien-Ju, 1279
 Ho, Wan Ching, 1117, 1119
 Hoey, Jesse, 963
 Hofmann, Lisa-Maria, 685
 HolmesParker, Chris, 1157
 Holvoet, Tom, 803
 Horvitz, Eric, 1089
 Howard, Steve, 1185
 Howley, Enda, 1211
 Hrstka, Ondřej, 1309
 Hsu, Jane Yung-Jen, 643, 1279
 Huang, Lixing, 1289

 Iba, Wayne, 1239
 Ichimura, Ryo, 1173
 Ienco, Dino, 1203
 Iocchi, Luca, 1327
 Iuliano, Claudio, 1125
 Iwasaki, Atsushi, 541, 651, 1173, 1269, 1271

 Jain, Manish, 327, 997, 1345
 Jakob, Michal, 989, 1273, 1309
 Jamroga, Wojciech, 727
 Janowski, Kathrin, 1093
 Jarquin, Roger, 1113
 Jayatilleke, Gaya, 285
 Jennings, Nicholas R., 5, 165, 363, 371, 787, 811, 1083, 1099, 1121
 Ji, Jianmin, 1301
 Joe, Yongjoon, 1269
 John, Richard, 1155
 Jonker, Catholijn M., 1231
 Julián, Vicente, 929, 1221
 Jumadinova, Janyl, 1217, 1361

 Köster, Michael, 1129
 Kafalı, Özgür, 1167, 1175
 Kaisers, Michael, 593, 1105, 1311
 Kalech, Meir, 115
 Kalyanakrishnan, Shivaram, 769
 Kamar, Ece, 1089
 Kamboj, Sachin, 13
 Kaminka, Gal A., 91, 115, 457
 Kash, Ian A., 175
 Katarzyniak, Radoslaw, 499
 Katsuragi, Atsushi, 541
 Kempton, Willett, 13
 Khalastchi, Eliahu, 115
 Khan, Shakil M., 1251
 Khosla, Pradeep, 753
 Kido, Hiroyuki, 267
 Kiekintveld, Christopher, 37, 997, 1005, 1155
 Kim, Yoonheui, 1153
 Kitaki, Makoto, 1271
 Kleiman, Elena, 525
 Knobbout, Max, 517
 Koenig, Sven, 123, 1069
 Kohli, Pushmeet, 1179, 1197
 Kolmogorov, Vladimir, 1197
 Kooi, Barteld, 711
 Korsah, G. Ayorkor, 1247
 Korzhyk, Dmytro, 327, 1013
 Kot, Alex C., 1151
 Kota, Ramachandra, 787, 1099
 Kowalczyk, Ryszard, 353, 499, 659, 1073
 Krainin, Michael, 1153
 Kraus, Sarit, 79, 345, 423, 567
 Kudenko, Daniel, 225, 1227
 Kuiper, Dane, 1235
 Kulis, Brian, 777
 Kumar, Akshat, 1087
 Kung, Jerry, 627
 Kunze, Lars, 107
 Kuo, Yen-Ling, 1279
 Kurihara, Satoshi, 233
 Kwak, Jun-Young, 1261

 Lützenberger, Marco, 1257, 1325
 López-Paz, David, 1315
 Lacerda, Bruno, 1253
 Lang, Jérôme, 79, 829
 Lang, Tobias, 1263
 Larson, Kate, 635, 1281
 La Poutré, Han, 1293
 Lee, Yew Ti, 1321
 Lemmens, Nyree, 1311
 Leo, Alberto, 1327
 Lespérance, Yves, 1251
 Lesser, Victor, 609, 1101, 1153, 1169
 Lev, Omer, 845

Le Borgne, Yann-Aël, 249
 Li, Guannan, 1113
 Li, H., 1303
 Li, Minyi, 353, 659, 1073
 Lim, Mei Yii, 1117, 1119
 Lima, Pedro U., 1253
 Lin, Andrew, 583
 Lin, Raz, 115
 Lipi, Afia Akhter, 441
 Lisý, Viliam, 989, 1273
 Littman, Michael, 593
 Liu, Siyuan, 1151
 Logan, Brian, 397
 Lohmann, Peter, 1129
 Longin, Dominique, 1031
 Lorini, Emiliano, 1031, 1207
 Lorkiewicz, Wojciech, 499
 Low, Kian Hsiang, 753
 Lund, Henrik Hautop, 1299
 Lupu, Emil, 1137
 Lv, Yanpeng, 1301

 Ma, Jiefei, 1137
 MacAlpine, Patrick, 769
 Maheswaran, Rajiv, 1313
 Manzoni, Sara, 1223
 Marcolino, Leandro Soriano, 21
 Marecki, Janusz, 1005
 Marengo, Elisa, 467
 Marsella, Stacy, 457, 1023
 Marsh, Stephen, 871
 Martin, Brent, 1113
 Martinho, Carlos, 1039
 Masuch, Nils, 1257
 Matsubara, Hitoshi, 21
 Maudet, Nicolas, 47
 McBurney, Peter, 879
 Meir, Reshef, 319
 Meneguzzi, Felipe, 1143, 1233
 Merrick, Kathryn E., 1067, 1075
 Meseguer, Pedro, 123, 379
 Meyer, John-Jules Ch., 301, 921
 Miao, Chunyan, 1151, 1159, 1169
 Michaely, Assaf, 319
 Michalak, Tomasz P., 1121
 Mihaylov, Mihail, 249
 Miller, Tim, 1185
 Modayil, Joseph, 761
 Mohite, Mayur, 1081
 Monnot, Jérôme, 829
 Morency, Louis-Philippe, 1289
 Moriyama, Koichi, 233
 Mouaddib, Abdel-Allah, 947
 Muñoz-Avila, Hector, 217

 Nakano, Yukiko, 441

 Narahari, Y., 1081
 Nardi, Daniele, 1327
 Nau, Dana, 337
 Navarro, Laurent, 701
 Nguyen, Nhung, 1047
 Nitta, Katsumi, 267
 Noam, Peled, 1363
 Noorian, Zeinab, 871
 Noot, Han, 1293
 Noriega, Pablo, 1191, 1305
 Norman, Timothy J., 913, 1233
 Nowé, Ann, 249, 1115
 Numao, Masayuki, 233

 Obraztsova, Svetlana, 71
 Ogden, Andrew, 457
 Ogston, Elth, 389
 Oh, Jean, 1233
 Ohtsuka, Kazumichi, 1223
 Okamoto, Steven, 1245
 Okaya, Masaru, 1297
 Okimoto, Tenda, 1269
 Onaindia, Eva, 971, 1195
 Ordonez, Fernando, 1155
 Ossowski, Sascha, 1099
 Owens, Sean, 1245

 Pěchouček, Michal, 327, 989, 1123, 1273, 1309
 Paay, Jeni, 1185
 Padget, Julian, 1109
 Padgham, Lin, 285
 Pagliarini, Luigi, 1299
 Paiva, Ana, 1039
 Pajares, Sergio, 971
 Panozzo, Fabio, 981
 Pardo, Pere, 971
 Pardoe, David, 887
 Parkes, David C., 627, 811
 Parr, Ronald, 1013
 Parsons, Simon, 879, 913, 1143
 Paruchuri, Praveen, 1165
 Patti, Viviana, 467
 Pedell, Sonja, 1185
 Pelachaud, Catherine, 1055
 Peled, Noam, 345
 Peleteiro, Ana, 669
 Perales, Francisco J., 1093
 Pesty, Sylvie, 1031
 Pfeffer, Avi, 1097
 Pigozzi, Gabriella, 1127, 1307
 Pilarski, Patrick M., 761
 Pini, Maria Silvia, 311, 1209
 Piras, Lena, 837
 Pita, James, 37, 1359
 Podlaszewski, Mikołaj, 1127, 1307
 Porteous, Julie, 449, 1323

Poupart, Pascal, 1133, 1263
 Prakken, H., 921
 Precup, Doina, 761, 1079
 Prepin, Ken, 1055
 Procaccia, Ariel D., 627
 Pujol-Gonzalez, Marc, 379
 Pulter, Natalja, 795
 Purvis, Martin, 1181

 Qu, Hongyang, 483

 Rückert, U., 1303
 Ramchurn, Sarvapali D., 5
 Ranathunga, Surangika, 1181
 Rebollo, M., 1241
 Rehak, Martin, 1123
 Rehm, Matthias, 441
 Restelli, Marcello, 199
 Rika, Inbal, 457
 Rivière, Jérémy, 1031
 Robu, Valentin, 787, 811
 Rodriguez, Inmaculada, 1131
 Rodriguez-Aguilar, Juan Antonio, 133, 379, 669
 Rogers, Alex, 5, 165, 363, 371, 787, 811
 Roos, Magnus, 853
 Rosenfeld, Avi, 423, 1177
 Rosenschein, Jeffrey S., 319, 845
 Rossi, Francesca, 311, 1209
 Rothe, Jörg, 837, 853
 Rovatsos, Michael, 1213, 1215
 Russo, Alessandra, 1137

 Sá, Samy, 1275, 1369
 Sánchez-Anguix, Víctor, 929, 1357
 Sabater-Mir, Jordi, 1161
 Salazar, Norman, 669
 Sandholm, Tuomas, 533, 1111
 Sapena, Oscar, 1195
 Sardina, Sebastian, 575
 Sarne, David, 415, 431
 Savani, Rahul, 559
 Scerri, Paul, 677, 955, 1245
 Schepperle, Heiko, 795
 Schindler, Ingo, 1199
 Schurr, Nathan, 1149
 Seedig, Hans Georg, 183
 Selman, Bart, 1277
 Sen, Sandip, 1161, 1239
 Serrano, Emilio, 1215
 Sha, Fei, 777
 Shafi, Kamran, 1075
 Shahidi, Neda, 1225
 Sheel, Ankur, 457
 Shen, Peijia, 1301
 Shen, Zhiqi, 1159, 1169
 Shieh, Eric, 133

 Shimura, Kenichiro, 1223
 Sierra, Carles, 1189, 1315
 Sietsma, Floor, 1183
 Sindlar, Michal, 301
 Singh, Munindar P., 293, 467, 475, 491, 863
 Singh, Satinder, 1287
 Sinn, Mathieu, 1133
 Sklar, Elizabeth, 879
 Slinko, Arkadii, 821
 Soh, Leen-Kiat, 1283
 Sollenberger, Derek J., 293
 Sombattheera, Chattrakul, 895
 Sonu, Ekhlas, 1259
 Stefanovitch, N., 363
 Steigerwald, Erin, 37
 Stein, Sebastian, 811
 Stentz, Anthony, 1247
 Sterling, Leon, 1185
 Stiborek, Jan, 1123
 Stone, Peter, 567, 769, 887, 1103, 1225
 Stranders, Ruben, 371
 Suay, Halit Bener, 617
 Such, Jose M., 1189, 1333
 Sugawara, Toshiharu, 1193
 Sujit, P. B., 1265
 Sun, Xiaoxun, 123, 1069
 Sutton, Richard S., 761
 Swarup, Samarth, 693
 Sycara, Katia, 677, 685, 955, 1143, 1165, 1233, 1245
 Szalas, Andrzej, 743

 Takahashi, Tomoichi, 1297
 Tambe, Milind, 37, 133, 327, 457, 997, 1005, 1155, 1261
 Tan, Ah-Hwee, 1159
 Tang, Yuqing, 879, 1143
 Tanoto, A., 1303
 Taylor, Matthew E., 457, 617, 777, 1261
 Tekbacak, Fatih, 1335
 Testa, Pietro, 1317
 Teutenberg, Jonathan, 449, 1323
 Thangarajah, John, 285, 1139
 Theng, Yin-Leng, 1151
 Thi Duong, Nguyen, 1147
 Todo, Taiki, 651
 Toni, Francesca, 1095, 1167
 Torreño, Alejandro, 1195
 Torroni, Paolo, 1167, 1175
 Toussaint, Marc, 1263
 Traskas, Dimitris, 1109
 Traub, Meytal, 91
 Tredan, Gilles, 1071
 Trescak, Tomas, 1131
 Troquard, Nicolas, 719
 Tsai, Jason, 457
 Tuglular, Tugkan, 1335

Turrini, Paolo, 727
Tuyls, Karl, 249, 551, 1105, 1311

Ueda, Suguru, 1173, 1271
Unland, Rainer, 1113
Urieli, Daniel, 769, 1103

Vaněk, Ondřej, 327, 1273, 1309, 1371
Vandael, Stijn, 803
van der Hoek, Wiebe, 149, 711, 719, 735, 1085
van der Torre, Leendert, 895, 1203
van der Weide, Tom L., 921
van Ditmarsch, Hans, 711
Van Dyke Parunak, H., 1077
van Eijck, Jan, 1183
van Oijen, Joost, 1249
van Riemsdijk, M. Birna, 405, 1231
Varakantham, Pradeep, 955, 1069, 1147, 1149
Vargas, Patricia A., 1117, 1119
Varona, Javier, 1093
Vasirani, Matteo, 1099
Velagapudi, Prasanna, 955
Venable, Kristen Brent, 311, 1209
Vesic, Srdjan, 1237
Vetere, Frank, 1185
Vikhorev, Konstantin, 397
Villatoro, Daniel, 1161, 1373
Vinyals, Meritxell, 133
Visser, Simeon, 1139
Vizzari, Giuseppe, 1223
Vo, Quoc Bao, 353, 499, 659, 1073
Vrancx, Peter, 1115
Vreeswijk, Gerard A.W., 517, 921
Vytelingum, Perukrishnen, 5

Wachsmuth, Ipke, 1047, 1135
Wagner, Hanno-Felix, 1113
Walsh, Toby, 311, 1209
Wang, Chongjun, 259
Wang, Dejian, 1301
Wang, Ko-Hsin Cindy, 1343
Wang, Shangfei, 1301
Wang, Wenjie, 1091
Wang, Xuezhi, 457
Waugh, Kevin, 1111
Weiss, Gerhard, 1311
Wenkstern, Rym Z., 1235
Werner, F., 1303
Westbrook, David, 609
Westra, Joost, 1291
White, Adam, 761
Wilson, Brandon, 337, 1355
Winikoff, Michael, 405, 1107, 1113
Witteveen, Cees, 149
Witwicki, Stefan J., 29
Wooldridge, Michael, 79, 149, 719, 735, 1083, 1085

Wu, Jun, 259
Wunder, Michael, 593

Xia, Lirong, 829
Xie, Junyuan, 259
Xin, Liu, 1071

Yadav, Nitin, 575
Yang, Qiang, 217
Yang, Rong, 1155, 1261
Yaros, John Robert, 593
Yeoh, William, 1069
Yin, Dong, 1301
Yin, Zhengyu, 133, 1261
Yoke Hean Low, Malcolm, 1321
Yokoo, Makoto, 541, 651, 1173, 1269, 1271
Young, Thomas, 1113
Yu, Han, 1159
Yu, Ling, 1169

Zeng, Fanchao, 1321
Zeng, Yifeng, 1229
Zhang, Haoqi, 627
Zhang, Jie, 1151
Zhang, Rong, 1301
Zhuo, Hankz Hankui, 217
Zick, Yair, 55
Ziebart, Brian D., 207
Zilberstein, Shlomo, 157, 1087
Zilka, Avishay, 457
Zink, Michael, 609
Zivan, Roie, 1145, 1165
Zuckerman, Inon, 337
Zuckerman, Michael, 845