AAMAS 2012

The 11th International Conference on Autonomous Agents and Multiagent Systems

June 4—8, 2012

Valencia, Spain

Proceedings

Volume I

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

ISBN-10: 0-9817381-1-7
ISBN-13: 978-0-9817381-1-6

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 high-profile and high-quality 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), Toronto (2010), and Taipei (2011). These are the proceedings of AAMAS 2012, held in Valencia, in June 2012.

In addition to the general track for the AAMAS 2012 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. The tracks were chaired by leaders in the corresponding fields: Daniele Nardi for the robotics track, Stefan Kopp for the virtual agents track, and Klaus Fischer and Alex Rogers 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 2012 Program Co-chairs in discussion with the special track chairs.

Only full paper submissions were solicited for AAMAS 2012. The general, robotics, virtual agents, and innovative applications tracks received 525, 75, 45, and 31 submissions respectively, for a total of 676 submissions at the submission deadline, with 671 papers going on to be reviewed (a few papers were withdrawn after the deadline).

After a thorough review and discussion process which included an opportunity for authors to respond to reviewer comments, 137 papers were selected for publication as full papers (acceptance rate 20.4%), each of which was allocated 8 pages in the proceedings and allocated 20 minutes in the program for oral presentation. Another 154 papers were selected as extended abstracts and allocated 2 pages each in the proceedings. Both full papers and extended abstracts were presented as posters during the conference. A number of accepted papers were subsequently withdrawn, leaving 136 full papers and 146 extended abstracts.

Of the submissions, 401 (59%) were indicated as being student papers, which indicates that AAMAS continues to be a nurturing environment for students. Submissions were assigned keywords, each of which was classified under one of 15 top-level topics (e.g., "Agent Cooperation"), including a new keyword "Perspectives" which attracted six submissions. Representation of top-level topics (measured by first keyword) was broad, with top counts in the areas of Economic Paradigms (113 submissions), Agent Cooperation (110), Learning and Adaptation (66), Agent Reasoning (64), Robotics (54) and Agreement Technologies (40). Looking at specific keywords (e.g., "Agent Cooperation::Distributed problem solving"), the most popular submission topics (again, measured by first keyword) were game theory (43 submissions), teamwork, coalition formation, and coordination (34), distributed problem solving (33), single agent learning (27), robot teams, multi-robot systems, robot coordination (26), planning (25), auction and mechanism design (24), and multiagent learning (21).

We thank the PC and SPC members of AAMAS 2012 for their thoughtful reviews and extensive discussions. We thank Daniele Nardi, Stefan Kopp, Klaus Fischer, and Alex Rogers for making the robotics, the virtual agents and the innovative applications tracks a success. We thank Mehdi Dastani and Dave Shield for putting together the proceedings. 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 Vicente Botti as Chair of the Local Organising Committee for his contributions. We also thank Dave Shield for his patience and support regarding Confmaster during every stage between the submission process and the actual AAMAS 2012 event.

Finally, we would like to thank the programme committee and senior programme committee members for their work, and the authors for submitting their work to AAMAS.

Vincent Conitzer and Michael Winikoff,
AAMAS 2012 Program Co-Chairs

Wiebe van der Hoek and Lin Padgham,
AAMAS 2012 General Co-Chairs

Organising Committee

General Co-Chairs

Lin Padgham (RMIT University, Australia)
Wiebe van der Hoek (University of Liverpool, UK)

Program Co-Chairs

Vincent Conitzer (Duke University, US)
Michael Winikoff (University of Otago, New Zealand)

Robotics Track Chair

Daniele Nardi (Sapienza University of Rome, Italy)

Virtual Agents Track Chair

Stefan Kopp (Bielefeld University, Germany)

Innovative Applications Track Chairs

Klaus Fischer (DFKI, Germany)
Alex Rogers (University of Southampton, UK)

Local Arrangements Chair

Vicent Botti (Universidad Politecnica de Valencia, Spain)

Finance Chair

Sascha Ossowski (University Rey Juan Carlos, Spain)

Publicity Chair

Stephen Cranefield (University of Otago, New Zealand)

Publications Chair

Mehdi Dastani (Utrecht University, Netherlands)

Tutorials Chairs

Christopher Kiekintveld (University of Texas at Elpaso, US)
Catherine Pelachaud (CNRS-LTC1 and Telecom ParisTech, France)

Workshops Chair

Elizabeth Sklar (City University of New York, US)

Exhibitions Chair

Karl Tuyls (Maastricht University, Netherlands)

Demonstrations Chair

Paul Scerri (Carnegie Mellon University, US)

Scholarships Co-Chairs

Maria Gini (University of Minnesota, US)
Janusz Marecki (IBM Research, US)
Michal Pěchouček (Czech Technical University, Czech Republic)

Doctoral Consortium Co-Chairs

Enrico Gerding (University of Southampton, UK)
Radhika Nagpal (Harvard University, US)

Sponsorship Co-Chairs

Virginia Dignum (Delft University of Technology, Netherlands)
Satoshi Kurihara (ISIR, Osaka University, Japan)
Sean Luke (George Mason University, US)

Senior Program Committee

Sherief Abdallah (British University in Dubai)
Thomas Ågotnes (University of Bergen)
Francesco Amigoni (Politecnico di Milano)
Elisabeth Andre (University of Augsburg)
Rafael Bordini (INF-UFRGS)
Craig Boutilier (University of Toronto)
Felix Brandt (TU Munich)
Monique Calisti (Martel Consulting)
Ruggiero Cavallo (Yahoo Research)
Xiaoping Chen (University of Science and Technology of China)
Yiling Chen (Harvard University)
Brad Clement (Jet Propulsion Laboratory)
Stephen Cranefield (University of Otago)
Sanmay Das (Rensselaer Polytechnic Institute)
Mathijs de Weerdt (Delft University of Technology)
Keith Decker (University of Delaware)
Frank Dignum (Utrecht University)
Ed Durfee (University of Michigan)
Piotr Faliszewski (AGH University of Science and Technology)
Boi Faltings (EPFL)
Ya'akov (Kobi) Gal (Harvard University)
Nicola Gatti (Politecnico di Milano)
Enrico Gerding (University of Southampton)
Maria Gini (University of Minnesota)
Dominic Greenwood (Whitestein Technologies)
Koen Hindriks (Delft University of Technology)
Michael Huhns (University of South Carolina)
Takayuki Ito (Nagoya Institute of Technology)
Catholijn Jonker (Delft University of Technology)
Gal Kaminka (The MAVERICK Group and Bar-Ilan University)
Jeffrey Kephart (IBM Research)
Christopher Kiekintveld (University of Texas at Elpaso)
Alexander Kleiner (Linköping University)
Sven Koenig (University of Southern California)
Nicole Kraemer (University Duisburg-Essen)
Sarit Kraus (Bar-Ilan University and University of Maryland)
Jérôme Lang (LAMSADE)
Kate Larson (University of Waterloo)
Yves Lespérance (York University)
James Lester (North Carolina State University)
Kevin Leyton-Brown (University of British Columbia)
Michael Luck (King's College London)

Sean Luke (George Mason University)
Rajiv Maheswaran (University of Southern California)
Janusz Marecki (IBM Research)
Stacy Marsella (University of Southern California)
Peter McBurney (King's College London)
Louis-Phillippe Morency (University of Southern California)
Yukiko Nakano (Seikei University)
Ana Paiva (INESC-ID and Instituto Superior Tecnico)
David Parkes (Harvard University)
Simon Parsons (City University of New York)
H. Van Dyke Parunak (Jacobs Vector Research Centre)
Adrian Pearce (University of Melbourne)
Michal Pěchouček (Czech Technical University)
Catherine Pelachaud (CNRS-LTC1 and Telecom ParisTech)
Ariel Procaccia (Carnegie Mellon University)
Sarvapali Ramchurn (University of Southampton)
Juan Antonio Rodriguez-Aguilar (IIIA-CSIC)
Alex Rogers (University of Southampton)
Francesca Rossi (University of Padova)
Tuomas Sandholm (Carnegie Mellon University)
Paul Scerri (Carnegie Mellon University)
Sandip Sen (University of Tulsa)
Onn Shehory (IBM Research)
Yoav Shoham (Stanford University)
Jaime Sichman (University of Sao Paulo)
Munindar Singh (North Carolina State University)
Elizabeth Sklar (City University of New York)
Liz Sonenberg (University of Melbourne)
Katia Sycara (Carnegie Mellon University)
Milind Tambe (University of Southern California)
Matthew Taylor (Lafayette College)
Moshe Tennenholtz (Microsoft Research and Technion)
John Thangarajah (RMIT University)
Simon Thompson (BT Research & Technology)
Kagan Tumer (Oregon State University)
Wamberto Vasconcelos (University of Aberdeen)
Manuela Veloso (Carnegie Mellon University)
Toby Walsh (NICTA and University of New South Wales)
Gerhard Weiss (University of Maastricht)
Danny Weyns (Linnaeus University, Campus Växjö)
Cees Witteveen (Delft University of Technology)
Makoto Yokoo (Kyushu University)
Pinar Yolum (Bogazici University)
Neil Yorke-Smith (American University of Beirut)
Shlomo Zilberstein (University of Massachusetts Amherst)

Program Committee

Noa Agmon (University of Texas at Austin)
Adrian Agogino (UCSC, NASA Ames Research Center)
Stephane Airiau (University of Amsterdam)
Marco Alberti (Universidade Nova de Lisboa)
Huib Aldewereld (Delft University of Technology)
Natasha Alechina (University of Nottingham)
Jan Allbeck (George Mason University)

Marty Allen (University of Wisconsin-La Crosse)
Christopher Amato (MIT)
Frédéric Amblard (IRIT-UT1)
Leila Amgoud (IRIT - CNRS)
Bo An (University of Southern California)
Elliot Anshelevich (Rensselaer Polytechnic Institute)
Alexander Artikis (NCSR Demokritos and Imperial College London)
Katie Atkinson (University of Liverpool)
Reyhan Aydogan (Delft University of Technology)
Ruth Aylett (Heriot-Watt University)
Haris Aziz (Technische Universität München)
Yoram Bachrach (Microsoft Research)
Matteo Baldoni (University of Torino)
Bikramjit Banerjee (University of Southern Mississippi)
Laura Barbulescu (CMU)
Cristina Baroglio (University of Torino)
Nicola Basilico (University of California, Merced)
Ana Bazzan (UFRGS)
Christian Becker-Asano (University of Freiburg)
Maren Bennewitz (University of Freiburg)
Jamal Bentahar (Concordia University)
Timothy Bickmore (Northeastern University)
Peter Biro (Hungarian Academy of Sciences)
Elizabeth Black (King's College London)
Jim Blythe (ISI, USC)
Olivier Boissier (Ecole des Mines de Saint-Etienne)
Tibor Bosse (Vrije Universiteit Amsterdam)
Luis Botelho (ISCTE-IUL)
Sylvain Bouveret (Grenoble INP - Ensimag and LIG)
Michael Bowling (University of Alberta)
Lars Braubach (University of Hamburg)
Frances Brazier (Delft University of Technology)
Cyril Brom (Charles University in Prague)
Nils Bulling (Clausthal University of Technology)
Juan C. Burguillo (University of Vigo)
Zack Butler (Rochester Institute of Technology)
Zhongtang Cai (Oracle Corp.)
Martin Caminada (University of Luxembourg)
Longbing Cao (University of Technology, Sydney)
Ioannis Caragiannis (University of Patras)
Stefano Carpin (University of California, Merced)
Cristiano Castelfranchi (ISTC-CNR and University of Siena)
Ginevra Castellano (University of Birmingham)
Marc Cavazza (University of Teesside)
Jesus Cerquides (IIIA-CSIC)
Brahim Chaib-draa (Laval University)
Tanmoy Chakraborty (Harvard University)
Georgios Chalkiadakis (Technical University of Crete)
Yu-Han Chang (University of Southern California)
Carlos Chesnevar (Universidad Nacional del Sur)
Maria Chli (Aston University)
Amit Chopra (University of Trento)
Helder Coelho (Universidade de Lisboa)
Robin Cohen (University of Waterloo)
Silvia Coradeschi (Orebro University)

Nikolaus Correll (University of Colorado, Boulder)
Célia da Costa Pereira (Université de Nice Sophia Antipolis)
Prithviraj Dasgupta (University of Nebraska at Omaha)
Mehdi Dastani (Utrecht University)
Patrick De Causmaecker (Katholieke Universiteit Leuven and KaHo Sint-Lieven)
Giuseppe De Giacomo (Sapienza University of Rome)
Steven de Jong (Maastricht University)
Tiago de Lima (University Lille Nord de France)
Francien Dechesne (Delft University of Technology)
James Delgrande (Simon Fraser University)
Scott DeLoach (Kansas State University)
Yves Demazeau (LIG-CNRS)
M. Bernardine Dias (Carnegie Mellon University)
Virginia Dignum (Delft University of Technology)
Oğuz Dikenelli (Ege University)
Klaus Dorer (Offenburg University)
Prashant Doshi (University of Georgia)
Barbara Dunin-Kęplicz (Warsaw University and Polsh Academy of Sciences)
Amal El Fallah Seghrouchni (LIP6 - UPMC and CNRS)
Edith Elkind (Nanyang Technological University)
Ulle Endriss (University of Amsterdam)
Gábor Erdélyi (Nanyang Technological University, Singapore)
Marc Esteva (IIIA-CSIC)
Rino Falcone (ISTC-CNR)
Alessandro Farinelli (University of Verona)
Maria Fasli (University of Essex)
Shaheen Fatima (Loughborough University)
Jacques Ferber (LIRMM - University of Montpellier 2)
Sevan Ficici (Natural Selection, Inc.)
Nicoletta Fornara (Universita della Svizzera Italiana)
Alex Fukunaga (The University of Tokyo)
Patrick Gebhard (DFKI)
Aditya Ghose (University of Wollongong)
Arpita Ghosh (Yahoo! Research)
Marco Gilles (Goldsmiths, University of London)
Andrew Gilpin (Hg Analytics)
Paolo Giorgini (University of Trento)
Piotr Gmytrasiewicz (University of Illinois at Chicago)
Claudia Goldman (GM Israel)
Valentin Goranko (Technical University of Denmark)
Guido Governatori (NICTA)
Gianluigi Greco (University of Calabria)
Rachel Greenstadt (Drexel University)
Nathan Griffiths (University of Warwick)
Davide Grossi (University of Liverpool)
Zahia Guessoum (Universite de Paris 6 and Universite de Reims)
Renata Guizzardi (Federal University of Espirito Santo)
Mingyu Guo (University of Liverpool)
Christian Guttmann (EBTIC)
James Hanson (IBM Research)
James Harland (RMIT University)
Paul Harrenstein (Technische Universität München)
Noam Hazon (Carnegie Mellon University)
Dirk Heylen (University of Twente)
Benjamin Hirsch (EBTIC)

Jesse Hoey (University of Waterloo)
Mark Hoogendoorn (VU University Amsterdam)
Ian Horswill (Northwestern University)
Adele Howe (Colorado State)
Jane Yung-jen Hsu (National Taiwan University)
Hung-Hsuan Huang (Ritsumeikan University)
Jomi Hubner (Federal University of Santa Catarina)
Joris Hulstijn (Delft University of Technology)
Wayne Iba (Westmont College)
Samuel Ieong (Microsoft Research)
Luca Iocchi (Sapienza University of Rome)
Atsushi Iwasaki (Kyushu University)
Michal Jakob (Czech Technical University)
Nadeem Jamali (University of Saskatchewan)
Wojciech Jamroga (University of Luxembourg)
Albert Xin Jiang (University of British Columbia and University of Southern California)
Kristiina Jokinen (University of Helsinki)
Patrick Jordan (Yahoo! Labs)
Radu Jurca (Google Inc)
Marcelo Kallmann (University of California, Merced)
Ece Kamar (MSR)
Sachin Kamboj (University of Delaware)
Kamalakar Karlapalem (IIIT-H)
Ian Kash (MSR UK)
Wolfgang Ketter (Erasmus University)
Tomas Klos (Delft University of Technology)
Franziska Klügl (Orebro University)
Matthias Klusch (DFKI)
Andreas Kolling (University of Pittsburgh)
Martin Kollingbaum (University of Aberdeen)
Sebastien Konieczny (CRIL-CNRS)
Andreas Krause (ETH Zurich)
Daniel Kudenko (University of York)
Han La Poutre (CWI and Universiteit Utrecht)
Michail Lagoudakis (Technical University of Crete)
Sebastien Lahaie (Yahoo! Research)
Luis Lamb (Universidade Federal do Rio Grande do Sul)
Brent Lance (Army Research Laboratory)
Alessandro Lazaric (SequeL)
João Leite (Universidade Nova de Lisboa)
Pedro Lima (Instituto Superior Tecnico)
Raz Lin (Bar-Ilan University)
Yaxin Liu (Google)
Brian Logan (University of Nottingham)
Alessio Lomuscio (Imperial College London)
Miguel Angel Lopez Carmona (Universidad de Alcala and MIT Sloan School of Management)
Maite Lopez-Sanchez (University of Barcelona)
Emiliano Lorini (IRIT)
Kian Hsiang Low (National University of Singapore)
Benjamin Lubin (Boston University)
Brian Magerko (Georgia Institute of Technology)
Roger Mailler (University of Tulsa)
Vangelis Markakis (Athens University of Economics and Business)
Carlos Martinho (Instituto Superior Tecnico and INESC-ID)
Viviana Mascardi (Universitádegli Studi di Genova)

Shigeo Matsubara (Kyoto University)
Tokuro Matsuo (Yamagata University)
Nicolas Maudet (LAMSADE, Univ. Paris-Dauphine)
Francisco Melo (INESC-ID/Instituto Superior Técnico)
Felipe Meneguzzi (Carnegie Mellon University)
Pedro Meseguer (IIIA CSIC)
John-Jules Meyer (Utrecht University and Alan Turing Institute Almere)
Tomasz Michalak (University of Southampton)
Martin Michalowski (Adventium Labs)
Simon Miles (King's College London)
Tim Miller (The University of Melbourne)
Sanjay Modgil (King's College London)
Luis Moniz (FCUL)
Pavlos Moraitis (LIPADE, Paris Descartes University)
David Morley (SRI International)
Abdel-Illah Mouaddib (University of Caen Basse-Normandie)
Joerg Mueller (TU Clausthal)
Rudolf Müller (Maastricht University)
David Musliner (SIFT)
Karen Myers (SRI)
Hideyuki Nakanishi (Osaka University)
Nanjangud Narendra (IBM)
Victor Naroditskiy (University of Southampton)
Abhaya Nayak (Macquarie University)
Radoslaw Niewiadomski (Telecom ParisTech)
Toyoaki Nishida (Kyoto University)
Jinzhong Niu (CUNY City College)
Timothy Norman (University of Aberdeen)
Ann Nowé (Vrije Universiteit Brussel)
Mariusz Nowostawski (University of Otago)
Colm O'Riordan (National University of Ireland)
Magalie Ochs (CNRS and Télécom ParisTech)
Frans Oliehoek (MIT)
Andrea Omicini (University of Bologna)
Nir Oren (University of Aberdeen)
Mehmet Orgun (Macquarie University)
Charlie Ortiz (SRI International)
Sascha Ossowski (University Rey Juan Carlos)
Eric Pacuit (Tilburg University)
Julian Padget (University of Bath)
Liviu Panait (Google)
Igor Pandzic (Zagreb University)
Mario Paolucci (ISTC-CNR)
David Pardoe (Yahoo! Labs)
Praveen Paruchuri (Carnegie Mellon University)
Terry Payne (University of Liverpool)
David Pennock (Yahoo! Research)
Anna Perini (FBK)
Christopher Peters (Coventry University)
Steve Phelps (University of Essex)
Paulo Pinheiro da Silva (University of Texas at El Paso)
Maria Silvia Pini (University of Padova)
Jeremy Pitt (Imperial College London)
Paul Piwek (The Open University)
Eric Platon (Cirius Technologies, Inc.)

Alexander Pokahr (University of Hamburg)
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)
Rui Prada (INESC-ID and Instituto Superior Técnico)
Henry Prakken (Utrecht University and University of Groningen)
Doina Precup (McGill University)
Maryam Purvis (University of Otago)
David Pynadath (University of Southern California)
Zinovi Rabinovich (Bar-Ilan University)
Iyad Rahwan (Masdar Institute and Massachusetts Institute of Technology)
Anita Raja (University of North Carolina at Charlotte)
Alessandro Ricci (University of Bologna)
Deborah Richards (Macquarie University)
Mark Riedl (Georgia Institute of Technology)
Laurel Riek (University of Notre Dame)
David Roberts (North Carolina State University)
David Robertson (University of Edinburgh)
Valentin Robu (University of Southampton)
Jeffrey Rosenschein (Hebrew University of Jerusalem)
Jörg Rothe (Heinrich-Heine-Universitaet Duesseldorf)
Antonino Rotolo (University of Bologna)
Michael Rovatsos (University of Edinburgh)
Wheeler Ruml (University of New Hampshire)
Zsófia Ruttkay (Moholy Nagy University of Art and Design)
Paul Rybski (Carnegie Mellon University)
Jordi Sabater-Mir (IIIA-CSIC)
Nicolas Sabouret (University Pierre et Marie Curie)
Fariba Sadri (Imperial College London)
Romeo Sanchez Nigenda (UANL)
Ken Satoh (National Institute of Informatics, Japan and Sokendai, Japan)
Bastin Tony Roy Savarimuthu (University of Otago)
Francesco Scarcello (University of Calabria)
Murat Şensoy (University of Aberdeen)
Sven Seuken (University of Zurich)
Guy Shani (Ben Gurion University)
Steven Shapiro (RMIT)
Alexei Sharpanskykh (VU University Amsterdam)
Dylan Shell (Texas A&M University)
Jiaying Shen (SRI International)
Mei Si (RPI)
Carles Sierra (IIIA-CSIC)
Guillermo Simari (Universidad Nacional del Sur in Bahia Blanca)
Gerardo Simari (Unversity of Oxford)
Arkadii Slinko (University of Auckland)
Stephen Smith (Carnegie Mellon University)
Leen-Kiat Soh (University of Nebraska)
Troels Sørensen (University of Warwick)
Tran Cao Son (New Mexico State University)
Matthijs Spaan (Delft University of Technology)
Siddharth Srivastava (University of Wisconsin Madison)
Mudhakar Srivatsa (IBM Research)
Sebastian Stein (University of Southampton)

Roni Stern (Ben Gurion University)
Nathan Sturtevant (University of Denver)
Gita Sukthankar (University of Central Florida)
Evan Sultanik (Johns Hopkins APL and Drexel University)
Samarth Swarup (Virginia Tech)
Pedro Szekely (USC Information Sciences Institute)
Erik Talvitie (Franklin & Marshall College)
Pingzhong Tang (Carnegie Mellon University)
Luke Teacy (University of Ulster)
Adriaan ter Mors (Delft University of Technology)
Andrea Tettamanzi (Universita degli Studi di Milano)
Michael Thielscher (The University of New South Wales)
Gian Diego Tipaldi (University of Freiburg)
Viviane Torres da Silva (Universidade Federal Fluminense)
Jan Treur (Vrije Universiteit Amsterdam)
Nicolas Troquard (ISTC-CNR)
Karl Tuyls (Maastricht University)
Leendert van der Torre (University of Luxembourg)
Hans van Ditmarsch (University of Sevilla)
Rogier van Eijk (Utrecht University)
Michael van Lent (SoarTech)
Peter-Paul van Maanen (TNO and Vrije Universiteit Amsterdam)
M. Birna van Riemsdijk (Delft University of Technology)
Greet Vanden Berghe (KaHo Sint-Lieven and K.U.Leuven)
Pradeep Varakantham (Singapore Management University)
Virginia Vassilevska Williams (University of California, Berkeley and Stanford University)
Kristen Brent Venable (University of Padova)
Mario Verdicchio (Università di Bergamo)
Paolo Viappiani (Aalborg University)
Hannes Vilhjalmsson (Reykjavik University)
Mirko Viroli (University of Bologna)
Bao Vo (Swinburne University of Technology)
Thomas Voice (University of Southampton)
Yevgeniy Vorobeychik (Sandia National Labs)
Peter Vrancx (Vrije Universiteit Brussel)
Marilyn Walker (UCSC)
Yonghong Wang (Carnegie Mellon University)
Michael Wellman (University of Michigan)
Shimon Whiteson (Univ of Amsterdam)
Mary-Anne Williams (University of Technology, Sydney)
Mark Wilson (University of Auckland)
Stefan Witwicki (INESC-ID and IST)
Wayne Wobcke (University of New South Wales)
Michael Wooldridge (University of Liverpool)
Lirong Xia (Harvard University)
Yang Xu (Univ. Electronic Science and Tech of China)
William Yeoh (Singapore Management University)
R. Michael Young (NC State University)
Minjie Zhang (University of Wollongong)
Chengqi Zhang (University of Technology, Sydney)
Yingqian Zhang (Erasmus University Rotterdam)
Martin Zinkevich (Yahoo! Labs)
Roie Zivan (Ben Gurion University)
Aviv Zohar (Microsoft Research)

Auxiliary Reviewers

Giulia Andrighetto
Chris Archibald
Itai Ashlagi
Edmond Awad
Dirk Bade
Aijun Bai
João Balsa
Nikhil Bansal
Nolan Bard
Titus Barik
Gregory J. Barlow
Dorothea Baumeister
Raphen Becker
Kostas Bekris
Alexandros-Sotiris Belesiotis
Aurélie Beynier
Graham Billiau
Darse Billings
Thomas Bolander
Branislav Bosansky
Fiemke Both
Maroua Bouzid
Simina Branzei
Keith Brawner
Darius Braziunas
Markus Brill
Brett Browning
Neil Burch
Chris Burnett
Ethan Burns
Xiaoqi Cao
Alan Carlin
Arthur Carvalho
Nilanjan Chakraborty
George Christelis
David Coleman
Florin Constantin
Matthew Crosby
Ricardo Matsumura de Araujo
Bart de Keijzer
Yann-Michaël De Hauwere
Enrique de la Hoz
Jeroen de Man
Christiano de Oliveira Braga
Francesco Maria Delle Fave
Sam Devlin
Jilles Dibangoye
John P. Dickerson
Yannis Dimopoulos
Ning Ding
John Doucette
Lachlan Dufton
Quang Duong
Felix Duvallet
Marcin Dziubinski
Adam Eck
Erdem Eser Ekinci
Mohamed El-Menshawy
Patricia Everaere
Paolo Felli
Ariel Felner
Jelena Fiosina
Maksims Fiosins
Amalia Foka
Antiono Franchi
Henry Franks
Katsuhide Fujita
Alfredo Gabaldon
Alice Gao
Hongxing Geng
Charlotte Gerritsen
Amineh Ghorbani
Robby Goetschalckx
Umberto Grandi
Alex Grigoriev
Tal Grinshpoun
Giorgio Grisetti
Marek Grzes
K. R. Guruprasad
Patricia Gutierrez
Sajjad Haider
Tayhun Gokmen Halac
Chung-Wei Hang
Brent Harrison
Ryan Harrison
Adnan Hashmi
Christopher Haubeck
Rafik Hedfi
Cédric Herpson
Rania Hodhod
Armin Hornung
Fatimah Modupe Ishowo-Oloko
Waqar Jaffry
Jie Jiang
Michael Johanson
Benjamin Johnston
E. Gil Jones
Janyl Jumadinova
Anshul Kanakia
Balajee Kannan
Patrick Kapahnke
Maria Karamitrou
Atif Khan
Shehroz Khan
Babak Khosravifar
Dominik Klein
Max Knobbout
Tomek Kolasa
Erik Komendera
David Kortenkamp
Andrew Koster
Ramachandra Kota
Annamaria Kovacs
Markus Kuderer
Tobias Kuester
Akshat Kumar
Jun-young Kwak
Marc Lanctot
Robert Lass
Ron Lavi
Matteo Leonetti
Josh Letchford
Omer Lev
Li Li
Minyi Li
Viliam Lisy
Angela Locoro
Robert Loftin
Marco Luetzenberger
Marin Lujak

Mahsa Maghami
Daniel Maier
Enrico Malizia
Marco Mamei
Cristina E. Manfredotti
Luca Marchetti
Ivan Marsa-Maestre
Maria Vanina Martinez
Riccardo De Masellis
James McInerney
João Messias
Gabriele Modena
Nataliya Mogles
Sara Montagna
Mirko Morandini
Hala Mostafa
Angelica Munoz-Melendez
Cu D. Nguyen
Hiroaki Nishi
José Nuno Pereira
Svetlana Obraztsova
Steven Okamoto
Fabio Paglieri
Luigi Palopoli
Thanos Panagopoulos
Fabio Patrizi
Noam Peled
Ana Peleteiro
Michael Pelican
Toni Penya
Markus Peters
Danilo Pianini
Cyril Poulet
Christos-Alexandros Psomas
Marc Pujol-Gonzalez
Fernando Velazquez Quesada
Srinivasa Ragavan
Dustin Reishus
Bryan Renne
Anja Rey
Reyhaneh Reyhani
Ariella Richardson
Emma Rollon
Magnus Roos
Jörg Röwekämper
Ji Ruan
Mike Ruberry
Zachary B. Rubinstein
Alex Rutherford
Jeff Rye
Leonardo Salayandia
David Sanderson
Monica Santos
Lena Schend
Claudio Schifanella
Pedro Sequeira
Oskar Skibski
Andrew Smith
Nikolaos I. Spanoudakis
Luciano Spinello
Christoph Sprunk
Isabelle Stanton
Alina Strachocka
Ashley Stroupe
Ken Sugawara
Andrzej Szalas
Charalampos Tampitsikas
Yuqing Tang
Danesh Tarapore
Jordan Thayer
Long Tran-Thanh
Pete Trautman
Paulo Trigo
Luca Tummolini
Paolo Turrini
Konstantina Valogianni
Janneke van der Zwaan
Harm van Seijen
Arlette van Wissen
Ondrej Vanek
Matteo Vasirani
Tiago Veiga
Matteo Venanzi
Sicco Verwer
Dani Villatoro
Meritxell Vinyals
Can Wang
Feng Wang
Yanjing Wang
Martijn Warnier
Stefan Warwas
Matthew Whitaker
Bryce Wiedenbeck
Colin Williams
Andreas Witzel
Feng Wu
Xiao-Feng Xie
Reda Yaich
Muhammad Yasir
Yifeng Zeng
Zongzhang Zhang
Xinghui Zhao
George Zhu
Ingo Zinnikus
Inon Zuckerman
Michael Zuckerman

Sponsors

We thank the following for their contribution to the success of this conference.

Emerald Sponsor

Platinum Sponsor

Gold Sponsor

Bronze Sponsor

Best Student Paper Sponsor

Scholarship Sponsors

Host Institutions

Contents

Invited Talks

Main Program - Full Papers

Session 1A – Innovative Applications

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

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

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

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

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

Session 2A – Virtual Agents

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

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

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

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

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

Session 3A – Robotics I

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

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

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

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

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

Session 4A – Robotics II

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

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

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

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

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

Session 5A – Robotics III

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

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

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

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

Session 1B – Teamwork I

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

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

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

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

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

Session 2B – Distributed Problem Solving

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

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

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

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

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

Session 4B – Agent Societies

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

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

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

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

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

Session 5B – Teamwork II

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

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

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

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

Session 1C – Learning I

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

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

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

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

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

Session 2C – Learning II

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

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

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

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

Session 3C – Human-agent Interaction

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

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

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

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

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

Session 4C – Argumentation & Negotiation

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

Cooperative Dialogues with Conditional Arguments
Samy Sá, João Alcântara 501

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

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

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

Session 5C – Emergence

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

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

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

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

Session 1D – Social Choice I

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

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

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

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

Session 2D – Social Choice II

Lot-based Voting Rules
Toby Walsh, Lirong Xia 603

Convergence of Iterative Voting
Omer Lev, Jeffrey Rosenschein 611

Optimal Manipulation of Voting Rules
Svetlana Obraztsova, Edith Elkind 619

Manipulation Under Voting Rule Uncertainty
Edith Elkind, Gábor Erdélyi 627

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

Session 3D – Economies & Markets I

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

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

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

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

Efficient Crowdsourcing Contests
Ruggiero Cavallo, Shaili Jain 677

Session 4D – Economies & Markets II

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

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

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

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

Fair Allocation Without Trade
Avital Gutman, Noam Nisan 719

Session 5D – Auction & Mechanism Design

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

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

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

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

Session 1E – Game Theory I

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

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

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

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

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

Session 2E – Game Theory II

Short Sight in Extensive Games
Davide Grossi, Paolo Turrini 805

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

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

Repeated zero-sum games with budget
Troels Sørensen 829

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

Session 3E – Game Theory III

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

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

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

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

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

Session 4E – Game Theory IV

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

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

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

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

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

Session 5E – Game & Agent Theories

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

Towards Tractable Boolean Games
Paul Dunne, Michael Wooldridge 939

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

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

Session 1F – Planning

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

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

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

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

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

Session 2F – Knowledge Representation & Reasoning

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

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

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

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

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

Session 3F – Agent-based Software Development

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

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

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

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

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

Session 4F – Logics for Agency

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

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

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

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

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

Session 5F – Logic and Verification

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

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

Semantics and Verification of Information-Based Protocols
Munindar Singh 1149

Main Program - Extended Abstracts

Innovative Applications

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

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

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

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

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

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

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

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

Virtual Agents

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

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

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

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

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

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

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

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

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

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

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

Robotics

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

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

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

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

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

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

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

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

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

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

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

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

Agent Reasoning

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

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

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

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

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

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

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

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

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

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

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

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

Coalitional Agency and Evidence-Based Ability
Nicolas Troquard 1245

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

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

Agent Cooperation

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

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

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

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

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

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

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

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

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

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

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

Global Optimization for Multiple Agents
Brammert Ottens, Boi Faltings 1273

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

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

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

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

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

Economic paradigms

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

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

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

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

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

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

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

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

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

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

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

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

Consensus Games
Julian Zappala, Natasha Alechina, Brian Logan 1309

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

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

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

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

TrustBets: Betting over an IOU Network
Sharad Goel, Mohammad Mahdian, David Pennock, Daniel Reeves 1319
On the Social Welfare of Mechanisms for Repeated Batch Matching
Elliot Anshelevich, Meenal Chhabra, Sanmay Das, Matthew Gerrior 1321
Merging Multiple Information Sources in Federated Sponsored Search Auctions
Sofia Ceppi, Enrico Gerding, Nicola Gatti 1323
A Truthful Learning Mechanism for Multi-Slot Sponsored Search Auctions with Externalities
Nicola Gatti, Alessandro Lazaric, Francesco Trovò 1325
Strategy-proof mechanisms for two-sided matching with minimum and maximum quotas
Suguru Ueda, Daniel Fragiadakis, Atsushi Iwasaki, Peter Troyan, Makoto Yokoo 1327
Incentives for Truthful Reporting in Crowdsourcing
Ece Kamar, Eric Horvitz 1329

Agent-based simulations

Cooperation among Malicious Agents: A General Quantitative Congestion Game Framework
Zaojie Rui, Tuanjie Fu, Darong Lai, Yichuan Jiang 1331
Opinion Convergence in Agent Networks
Sreerupa Chatterjee, Alexander Ruff, Sandip Sen 1333
Behavior Modeling From Learning Agents: Sensitivity to Objective Function Details
Robert Junges, Franziska Klügl 1335
Emergent Behaviour of Bacteria in a Multiagent System
Philip Hendrix, Elena Budrene, Benoit Morel, Igor Linkov 1337
Investigating the Role of Social Behavior in Financial Markets through Agent-Based Simulation
Alessia Mauri, Andrea Tettamanzi 1339
An Agent-Based Model for Pedestrian and Group Dynamics: Experimental and Real-World Scenarios
Giuseppe Vizzari, Lorenza Manenti 1341
The Impact of Cultural Differences on Crowd Dynamics
Natalie Fridman, Avishay Zilka, Gal Kaminka 1343
The Spanish Steps flower scam - agent-based modeling of a complex social interaction
Ladislau Bölöni 1345
Effect of defectors for cooperation: How strictly should defectors be eliminated from the newcomers?
Hitoshi Yamamoto, Isamu Okada, Yuki Ogawa 1347
Patterns of Migration and Adoption of Choices By Agents in Communities
Feyza Hafizoğlu, Sandip Sen 1349
Agent-based simulation of mobility in real-world transportation networks
Maicon Amarante, Ana Bazzan 1351
SimAnalyzer: Automated description of groups dynamics in agent-based simulations
Philippe Caillou, Javier Gil-Quijano 1353

Agent societies and Societal issues

Emergence of Cooperation through Structural Changes and Incentives in Service-Oriented MAS
Elena del Val, Miguel Rebollo, Vicent Botti 1355
Disagreement for control of rational cheating in peer review: a simulation
Mario Paolucci, Francisco Grimaldo 1357
Sub-delegation and Trust
Chris Burnett, Nir Oren 1359
A Dempster-Shafer Theory Based Witness Trustworthiness Model
Siyuan Liu, Alex C. Kot, Chunyan Miao, Yin-Leng Theng 1361
Detecting and Identifying Coalitions
Reid Kerr, Robin Cohen 1363

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

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

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

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

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

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

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

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

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

Learning and Adaptation

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

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

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

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

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

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

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

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

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

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

Agreement Technologies

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

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

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

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

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

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

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

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

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

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

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

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

Systems and Organisation

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

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

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

Agent-based system development

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

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

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

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

Agent theories - Models and Architectures

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

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

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

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

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

Demonstrations

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

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

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

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

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

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

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

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

Pogamut Toolkit
Jakub Gemrot, Michal Bída, Cyril Brom 1469

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Invited Talks

Delivering the Smart Grid:
A Grand Challenge for Autonomous Agents Research

Restructuring electricity grids to meet the increased demand of electric vehicles and heat pumps, while making greater use of intermittent renewable energy sources, represents one of the greatest engineering challenges of our day. This modern electricity grid, in which both electricity and information flow in two directions between large numbers of widely distributed suppliers and generators - commonly termed the 'smart grid' - represents a radical reengineering of infrastructure which has changed little over the last hundred years. However, the autonomous behaviour expected of the smart grid, its highly distributed nature, and the existence of multiple stakeholders each with their own incentives and interests, challenges existing engineering approaches. In this talk, I will describe why I believe that autonomous agents and multi-agent systems are essential for delivering the smart grid as it is envisioned. I will present some recent work that has been done in this area, and describe many challenges that still remain.

Alex Rogers *(University of Southampton)*

Alex Rogers is a Reader in the Agents, Interaction and Complexity Research Group at the University of Southampton in the UK. Originally graduating with a degree in Physics, he spend five years working as a field engineer in the oil industry before returning to academia having developed an interest in complexity science and multi-agent systems. His research interests address the challenges in developing and applying agent-based algorithms and mechanisms for the control of decentralised systems.

This work has addressed applications in areas such as sensor networks and unmanned autonomous vehicles, and most recently, has focused on applications within future energy systems such as the smart grid.

Lab and field evidence of a cognitive hierarchy in strategic thinking

When software agents interact with people, game theory provides a framework to help the agents make decisions. However, human behavior in games differs from that of the infinitely rational beings studied in classical game theory. Cognitive hierarchy (CH) models offer an algorithmic approach to modelling bounded rationality in strategic thinking, particularly for new strategic environments or as initial conditions for models of learning from experience. CH models have been applied to many experimental data sets, and to some field settings including Swedish lottery games and quality disclosure of movies through critics' reviews. There is also evidence from measuring visual attention, and fMRI of brain activity, which is consistent with steps of strategic thinking.

Colin Camerer *(California Institute of Technology)*

Colin Camerer is the Robert Kirby Professor of Behavioral Economics at Caltech. He earned a Ph.D. from the University of Chicago in 1981 and worked at Northwestern, Penn, and Chicago before Caltech. He has published more than 150 peer-reviewed articles and book chapters and wrote or co-edited four books. Camerer's research group is interested in the psychological and neural basis of choice, strategizing in games, and trading in markets.

Our focus is on complex goal-directed choices which typically involve rewards that depend on random events or choices by others. Recent neuroeconomic fMRI projects involve self-control in choosing tempting foods, weighting probabilities, curiosity, choice overload, and the contrast between hypothetical and binding (real) choices. His group also does economics using field data-testing game theory models of realistic limits on strategic thinking, using Swedish lotteries and movie revenues. Earlier projects examine hot hand misperceptions and sunk cost fallacies in NBA basketball, and labor supply of cab drivers. Our group also does field experiments, studying risk and time preferences, and group favoritism in Vietnam. Prof. Camerer has been the past president of the Economic Science (experimental economics) Association and the Society for Neuroeconomics, and was elected a Fellow of the Econometric Society and a member of the American Academy of Arts and Sciences.

Social Contexts

This talk will advocate the explicit treatment of social contexts for the design of automated agents and multi-agent systems. In particular, I will illustrate how social contexts effect the design of optimization algorithms, how social contexts can be designed to lead to efficient and stable multi-agent systems, and how adopting assumptions about the nature of the social context can provide powerful solutions to classical challenges in game theory and reinforcement learning.

Moshe Tennenholtz *(Technion/Microsoft Research Israel)*
2012 ACM/SIGART Autonomous Agents research award winner

Moshe Tennenholtz is the Sonheimer Professor at the Technion–Israel Institute of Technology. He is also a Principal Researcher at Microsoft Research and a founder of the basic research group at the Microsoft Israel R&D center. Moshe received his B.Sc. in Mathematics from Tel-Aviv University (1986), and his M.Sc. and Ph.D. (1987, 1991) from the Department of Applied Mathematics and Computer Science in the Weizmann Institute.

Moshe served as the editor-in-chief of the Journal of Artificial Intelligence Research [JAIR]; he is also an associate editor of Games and Economic Behavior, the international journal of autonomous agents and multi-agent systems, and of the transactions on economics and computation, serves on the editorial board of the Journal of Machine Learning Research, the moderator for the computer science and game theory section of the arXiv, and served on the editorial board of the AI magazine.

Moshe is a AAAI fellow and a fellow of the society for advancement of economic theory. He served as program chair of the ACM Electronic Commerce [EC] conference, and of the TARK conference. He was also co-founder and chief scientist of companies in the area of e-commerce. In joint work with colleagues and students he introduced several pioneering contributions to the interplay between computer science and game theory, such as the study of artificial social systems, co-learning, non-cooperative computing, distributed games, the axiomatic approach to qualitative decision making, the axiomatic approach to ranking, reputation, and trust systems, competitive safety analysis, program equilibrium, mediated equilibrium, and learning equilibrium, as well as the first near-optimal polynomial algorithm for reinforcement learning in stochastic games.

Social Norms for Self-Policing Multi-Agent Systems and Virtual Societies

Social norms help people self-organizing in many situations where having an authority representative is not feasible. On the contrary to institutional rules, the responsibility to enforce social norms is not the task of a central authority but a task of each member of the society. In recent years, the use of social norms has been considered also as a mechanism to regulate virtual societies and specifically heterogeneous societies formed by humans and artificial agents.

Firstly we sketch a game-theoretical categorization of norms that will organize the rest of the talk. This dissertation generally tackles how norms (assuming their existence) become established inside a virtual society, such as those formed entirely by virtual agents or a combination of them with human subjects. We initially tackle how conventions emerge when dealing with different topological structures of interactions. In this part we discovered how in social networks (with the theoretical characteristics of a scale-free) conventions cannot always emerge (even in the self-interest of the whole society), because of the emergence of subconventions that are facilitated by the inherent structure of the network. The identification of the Self-Reinforcing Substructures have allowed us to develop the necessary mechanisms to reach full convergence, which was never previously reached by any other researcher in the community.

After that we explore other mechanisms that allow the imposition of social norms, such as incentives mechanisms like punishment. We present an empirical study of how different punishment technologies affect differently human subjects and we develop an agent architecture (EMIL-I-A) which behaves similarly. This architecture is not only affected by the costs associated to punishment but also by the normative message it conveys, allowing the transmission of normative messages, establishing therefore the differentiation between punishment and sanction. This hypothesis is tested using a cross-methodological approach performing human experimentation and agent based simulation.

Finally, we explore another cognitive mechanism that would allow us to explain the voluntary non self-interested compliance, Internalization, by which agents comply with norms because so doing is an end in itself, and not merely because of external sanctions, such as material rewards or punishment.

Daniel Villatoro *(Autonomous University of Barcelona, Spain)*
2011 Victor Lesser Distinguished Dissertation award winner

Daniel Villatoro completed his PhD at the IIIA-CSIC under the supervision of Dr. Jordi Sabater-Mir. His main research interests focus on self-policing mechanisms for the adaptation of virtual environments, paying special attention to the interaction of virtual entities and human subjects. He has collaborated with known researchers in the area such as Sandip Sen, Rosaria Conte, Giulia Andrighetto or Michael Luck, and visited important institutions such as the Santa Fe Institute. Daniel has over 20 publications in top tier conferences and specialized journals. Moreover he has been an active member of the community acting as general chair of the EASSS09 and EASSS11, and the MABS11 Workshop, and reviewer of the most important journals (such as JAAMAS, EAAI, or ACM TAAS) and conferences (such as AAAI, IJCAI, AAMAS or ECAI).

A market-oriented programming enviornment and its application to distributed multicommodity flow problems

(Journal of Artificial Intelligence Research, Volume 1, pages 1-23, 1993)

Market price systems constitute a well-understood class of mechanisms that under certain conditions provide effective decentralization of decision making with minimal communication overhead. In a *market-oriented programming* approach to distributed problem solving, we derive the activities and resource allocations for a set of computational agents by computing the competitive equilibrium of an artificial economy. WALRAS provides basic constructs for defining computational market structures, and protocols for deriving their corresponding price equilibria. In a particular realization of this approach for a form of multicommodity flow problem, we see that careful construction of the decision process according to economic principles can lead to efficient distributed resource allocation, and that the behavior of the system can be meaningfully analyzed in economic terms.

Michael P. Wellman *(University of Michigan)*

2012 IFAAMAS Award for Influential Papers in Autonomous Agents and Multiagent Systems winner

Michael P. Wellman is Professor of Computer Science & Engineering at the University of Michigan.

He received a PhD from the Massachusetts Institute of Technology in 1988 for his work in qualitative probabilistic reasoning and decision-theoretic planning. From 1988 to 1992, Wellman conducted research in these areas at the USAF's Wright Laboratory. For the past 19+ years, his research has focused on computational market mechanisms for distributed decision making and electronic commerce.

As Chief Market Technologist for TradingDynamics, Inc. (now part of Ariba), he designed configurable auction technology for dynamic business-to-business commerce. Wellman previously served as Chair of the ACM Special Interest Group on Electronic Commerce (SIGecom), and as Executive Editor of the Journal of Artificial Intelligence Research. He is a Fellow of the Association for the Advancement of Artificial Intelligence and the Association for Computing Machinery.

Towards Flexible Teamwork

(Journal of Artificial Intelligence Research, Volume 7, pages 83-124, 1997)

Many AI researchers are today striving to build agent teams for complex, dynamic multi-agent domains, with intended applications in arenas such as education, training, entertainment, information integration, and collective robotics. Unfortunately, uncertainties in these complex, dynamic domains obstruct coherent teamwork. In particular, team members often encounter differing, incomplete, and possibly inconsistent views of their environment. Furthermore, team members can unexpectedly fail in fulfilling responsibilities or discover unexpected opportunities. Highly flexible coordination and communication is key in addressing such uncertainties. Simply fitting individual agents with precomputed coordination plans will not do, for their inflexibility can cause severe failures in teamwork, and their domain-specificity hinders reusability.

Our central hypothesis is that the key to such flexibility and reusability is providing agents with general models of teamwork. Agents exploit such models to autonomously reason about coordination and communication, providing requisite flexibility. Furthermore, the models enable reuse across domains, both saving implementation effort and enforcing consistency. This article presents one general, implemented model of teamwork, called STEAM. The basic building block of teamwork in STEAM is joint intentions (Cohen & Levesque, 1991b); teamwork in STEAM is based on agents' building up a (partial) hierarchy of joint intentions (this hierarchy is seen to parallel Grosz & Kraus's partial SharedPlans, 1996). Furthermore, in STEAM, team members monitor the team's and individual members' performance, reorganizing the team as necessary. Finally, decision-theoretic communication selectivity in STEAM ensures reduction in communication overheads of teamwork, with appropriate sensitivity to the environmental conditions. This article describes STEAM's application in three different complex domains, and presents detailed empirical results.

Milind Tambe *(University of Southern California)*
2012 IFAAMAS Award for Influential Papers in Autonomous Agents and Multiagent Systems winner

Milind Tambe is a Professor of Computer Science and Industrial and Systems Engineering at the University of Southern California(USC). He leads the TEAMCORE Research Group at USC, with research focused on agent-based and multi-agent systems. He is a fellow of AAAI (Association for Advancement of Artificial Intelligence) and recipient of the ACM (Association for Computing Machinery) "Autonomous Agents Research Award".

He is also the recipient of the Christopher Columbus Fellowship Foundation Homeland security award, the Rist Prize of the Military Operations Research Society, a "most influential paper award" from the International Foundation for Agents and Multiagent Systems, US First Coast Guard District's Operational Excellence Award, Certificate of Appreciation from the US Federal Air Marshals Service, special commendation given by the Los Angeles World Airports police from the city of Los Angeles, USC Viterbi School of Engineering use-inspired research award, Okawa foundation faculty research award, the RoboCup scientific challenge award, USC Steven B. Sample Teaching and Mentoring award and the ACM recognition of service award.

Prof. Tambe and his research group's papers have been selected as best papers at a dozen premier Artificial Intelligence and Operations Research Conferences and workshops; these have included best paper awards at the International Conference on Autonomous Agents and Multiagent Systems and International Conference on Intelligent Virtual Agents. Additionally, algorithms developed by his Teamcore research group have been deployed for real-world use by several agencies including the LAX police, the Federal Air Marshals service, the US Coast Guard and the Transportation security administration. He received his Ph.D. from the School of Computer Science at Carnegie Mellon University.

Main Program - Full Papers

Session 1A
Innovative Applications

PROTECT: A Deployed Game Theoretic System to Protect the Ports of the United States

Eric Shieh[+], Bo An[+], Rong Yang[+], Milind Tambe[+], Craig Baldwin*, Joseph DiRenzo*, Ben Maule*, Garrett Meyer*
[+]University of Southern California
[+]{eshieh, boa, yangrong, tambe}@usc.edu
*United States Coast Guard
*{Craig.W.Baldwin, Joseph.DiRenzo, Ben.J.Maule, Garrett.R.Meyer}@uscg.mil

ABSTRACT

While three deployed applications of game theory for security have recently been reported at AAMAS [12], we as a community remain in the early stages of these deployments; there is a continuing need to understand the core principles for innovative security applications of game theory. Towards that end, this paper presents PROTECT, a game-theoretic system deployed by the United States Coast Guard (USCG) in the port of Boston for scheduling their patrols. USCG has termed the deployment of PROTECT in Boston a success, and efforts are underway to test it in the port of New York, with the potential for nationwide deployment.

PROTECT is premised on an attacker-defender Stackelberg game model and offers five key innovations. First, this system is a departure from the assumption of perfect adversary rationality noted in previous work, relying instead on a quantal response (QR) model of the adversary's behavior — to the best of our knowledge, this is the first real-world deployment of the QR model. Second, to improve PROTECT's efficiency, we generate a compact representation of the defender's strategy space, exploiting equivalence and dominance. Third, we show how to practically model a real maritime patrolling problem as a Stackelberg game. Fourth, our experimental results illustrate that PROTECT's QR model more robustly handles real-world uncertainties than a perfect rationality model. Finally, in evaluating PROTECT, this paper for the first time provides real-world data: (i) comparison of human-generated vs PROTECT security schedules, and (ii) results from an Adversarial Perspective Team's (human mock attackers) analysis.

Categories and Subject Descriptors

J.m [**Computer Applications**]: MISCELLANEOUS

General Terms

Security, Design

Keywords

Game Theory, Security, Applications, Stackelberg Games

1. INTRODUCTION

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

The global need for security of key infrastructure with limited resources has led to significant interest in research conducted in multiagent systems towards game-theory for real-world security. As reported previously at AAMAS, three applications based on Stackelberg games have been transitioned to real-world deployment. This includes ARMOR, used by the Los Angeles International Airport [12] to randomize checkpoints of roadways and canine patrols; IRIS which helps the US Federal Air Marshal Service [12] in scheduling air marshals on international flights; and GUARDS [12] which is under evaluation by the US Transportation Security Administration to allocate resources for airport protection. We as a community remain in the early stages of these deployments, and must continue to develop our understanding of core principles of innovative applications of game theory for security.

To this end, this paper presents a new game-theoretic security application to aid the United States Coast Guard (USCG), called *Port Resilience Operational/Tactical Enforcement to Combat Terrorism* (PROTECT). The USCG's mission includes maritime security of the US coasts, ports, and inland waterways; a security domain that faces increased risks in the context of threats such as terrorism and drug trafficking. Given a particular port and the variety of critical infrastructure that an adversary may attack within the port, USCG conducts patrols to protect this infrastructure; however, while the adversary has the opportunity to observe patrol patterns, limited security resources imply that USCG patrols cannot be at every location 24/7. To assist the USCG in allocating its patrolling resources, similar to previous applications [12], PROTECT uses an attacker-defender Stackelberg game framework, with USCG as the defender against terrorist adversaries that conduct surveillance before potentially launching an attack. PROTECT's solution is to typically provide a mixed strategy, i.e. randomized patrol patterns taking into account the importance of different targets, and the adversary's surveillance and anticipated reaction to USCG patrols.

While PROTECT builds on previous work, this paper highlights five key innovations. The first and most important is PROTECT's departure from the assumption of perfect rationality on the part of the human adversaries. While appropriate in the initial applications as a first step — ARMOR, IRIS, GUARDS — this assumption of perfect rationality is well-recognized as a limitation of classical game theory, and bounded rationality has received significant attention in behavioral game-theoretic approaches [4]. Within this behavioral framework, quantal response equilibrium has emerged as a promising approach to model human bounded rationality [4, 10, 14] including recent results illustrating the benefits of the quantal response (QR) model in security games contexts [15]. Therefore, PROTECT uses a novel algorithm called PASAQ [16] based on the QR model of a human adversary. To the best of our knowledge, this

is the *first time that the QR model has been used in a real-world security application*.

Second, PROTECT improves PASAQ's efficiency via a compact representation of defender strategies exploiting dominance and equivalence analysis. Experimental results show the significant benefits of this compact representation. Third, PROTECT addresses practical concerns of modeling real-world maritime patrolling application in a Stackelberg framework. Fourth, this paper presents a detailed simulation analysis of PROTECT's robustness to uncertainty that may arise in the real-world. For various cases of added uncertainty, the paper shows that PROTECT's quantal-response-based approach leads to significantly improved robustness when compared to an approach that assumes full attacker rationality.

PROTECT has been in use at the port of Boston since April 2011 and been evaluated by the USCG. This evaluation brings forth our final key contribution: for the first time, this paper provides real-world data comparing human-generated and game-theoretic schedules. We also provide results from an Adversarial Perspective Team's (APT) analysis and comparison of patrols before and after the use of the PROTECT system from a viewpoint of an attacker. Given the success of PROTECT in Boston, we are now extending it to the port of New York, and based on the outcome there, it may potentially be extended to other ports in the US.

(a) PROTECT is being used in Boston

(b) Extending PROTECT to NY

Figure 1: USCG boats patrolling the ports of Boston and NY

2. USCG AND PROTECT'S GOALS

The USCG continues to face challenges with evolving asymmetric threats within the maritime environment not only within the Maritime Global Commons, but also within the ports and waterways that make up the United States Maritime Transportation System. The former Director of National Intelligence, Dennis Blair noted in 2010 a persistent threat "from al-Qa'ida and potentially others who share its anti-Western ideology. A major terrorist attack may emanate from either outside or inside the United States" [3]. This threat was reinforced in May of 2011 following the raid on Osama Bin Laden's home, where a large trove of material was uncovered, including plans to attack an oil tanker. "There is an indication of intent, with operatives seeking the size and construction of tankers, and concluding it's best to blow them up from the inside because of the strength of their hulls" [6]. These oil tankers transit the U.S. Maritime Transportation System. The USCG plays a key role in the security of this system and the protection of seaports to support the economy, environment, and way of life in the US.

Coupled with challenging economic times, USCG must operate as effectively as possible, achieving maximum benefit from every hour spent on patrol. As a result, USCG is compelled to re-examine the role that optimization of security resource usage plays in its mission planning — and how innovation provided by game theory can be effectively employed.

The goal of PROTECT is to use game theory to assist the USCG in maximizing its effectiveness in the Ports, Waterways, and Coastal Security (PWCS) Mission. PWCS patrols are focused on protecting critical infrastructure; without the resources to provide one hundred percent on scene presence at any, let alone all of the critical infrastructure, optimization of security resource is critical. Towards that end, unpredictability creates situations of uncertainty for an enemy and can be enough to deem a target less appealing.

The PROTECT system, focused on the PWCS patrols, addresses how the USCG should optimally patrol critical infrastructure in a port to maximize protection, knowing that the adversary may conduct surveillance and then launch an attack. While randomizing patrol patterns is key, PROTECT also addresses the fact that the targets are of unequal value, understanding that the adversary will adapt to whatever patrol patterns USCG conducts. The output of PROTECT is a schedule of patrols which includes when the patrols are to begin, what critical infrastructure to visit for each patrol, and what activities to perform at each critical infrastructure. While initially pilot tested in the port of Boston, the solution technique was intended to be generalizable and applicable to other ports.

3. KEY INNOVATIONS IN PROTECT

The PWCS patrol problem was modeled as a leader-follower (or attacker-defender) Stackelberg game [7] with USCG as the leader (defender) and the terrorist adversaries in the role of the follower. The choice of this framework was supported by prior successful applications of Stackelberg games [12]. In this Stackelberg game framework, the defender commits to a mixed (randomized) strategy of patrols, whereas the attacker conducts surveillance of these mixed strategies and responds with a pure strategy of an attack on a target. The objective of this framework is to find the optimal mixed strategy for the defender.

Stackelberg games have been well established in the multi-agent systems literature [5, 8, 9, 12]. Therefore, rather than providing further background in these games, this section immediately transitions to three of PROTECT's key innovations. We begin by discussing how to practically cast this real-world maritime patrolling problem of PWCS patrols as a Stackelberg game (Section 3.1). We also show how to reduce the number of defender strategies (Section 3.2) before addressing the most important of the innovations in PROTECT: its use of the quantal response model (Section 3.3).

3.1 Game Modeling

To model the USCG patrolling domain as a Stackelberg game, we need to define (i) the set of attacker strategies, (ii) the set of defender strategies, and (iii) the payoff function. These strategies and payoffs center on the targets in a port — ports, such as the port of Boston, have a significant number of potential targets (critical infrastructure). In our Stackelberg game formulation, the attacker conducts surveillance on the mixed strategies that the defender has committed to, and can then launch an attack. Thus, the attacks an attacker can launch on different possible targets are considered as his/her pure strategies.

However, the definition of defender strategies is not as straightforward. Patrols last for some fixed duration during the day as specified by USCG, e.g. 4 hours. Our first attempt was to model each target as a node in a graph and allow patrol paths to go from each individual target to (almost all) other targets in the port, generating an almost complete graph on the targets. This method yields the most flexible set of patrol routes that would fit within the maximum duration, covering any permutation of targets within a single patrol.

This method unfortunately faced significant challenges: (i) it required determining the travel time for a patrol boat for each pair of targets, a daunting knowledge acquisition task given the hundreds of pairs of targets; (ii) it did not maximize the use of port geography whereby boat crews could observe multiple targets at once and; (iii) it was perceived as micromanaging the activities of the USCG boat crews, which was undesirable.

Our improved approach to generating defender strategies therefore grouped nearby targets into patrol areas. The presence of patrol areas led the USCG to redefine the set of defensive activities to be performed on patrol areas to provide a more accurate and expressive model of the patrols. Activities that take a longer time provide the defender a higher payoff compared to activities that take a shorter time to complete. This impacts the final patrol schedule as one patrol may visit fewer areas but conduct longer duration defensive activities at the areas, while another patrol may have more areas with shorter duration activities.

To generate all the permutations of patrol schedules, a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is created with the patrol areas as vertices $\mathcal{V}$ and adjacent patrol areas as edges $\mathcal{E}$. Using the graph of patrol areas, PROTECT generates all possible patrol schedules, each of which is a closed walk of $\mathcal{G}$ that starts and ends at the patrol area $b \in \mathcal{V}$, the base patrol area for the USCG. The patrol schedules are a sequence of patrol areas and associated defensive activities, and are constrained by a maximum patrol time τ.

The graph $\mathcal{G}$ along with the constraints b and τ are used to generate the defender strategies (patrol schedules). Given each patrol schedule, the total patrol schedule time is calculated (this also includes traversal time between areas, but we ignore it in the following for expository purposes); we then verify that the total time is less than or equal to the maximum patrol time τ. After generating all possible patrol schedules, a game is formed where the set of defender strategies is composed of patrol schedules and the set of attacker strategies is the set of targets. The attacker's strategy was based on targets instead of patrol areas because an attacker will choose to attack a single target.

Table 1 gives an example, where the rows correspond to the defender's strategies and the columns correspond to the attacker's strategies. In this example, there are two possible defensive activities, activity k_1 and k_2, where k_2 provides a higher payoff for the defender than k_1. Suppose that the time bound disallows more than two k_2 activities (given the time required for k_2) within a patrol. Patrol area 1 has two targets (target 1 and 2) while patrol areas 2 and 3 each have one target (target 3 and 4 respectively). In the table, a patrol schedule is composed of a sequence of patrol areas and a defensive activity in each area. The patrol schedules are ordered so that the first patrol area in the schedule denotes which patrol area the defender needs to visit first. In this example, patrol area 1 is the base patrol area, and all of the patrol schedules begin and end at patrol area 1. For example, the patrol schedule in row 2 first visits patrol area 1 with activity k_2, then travels to patrol area 2 with activity k_1, and returns back to patrol area 1 with activity k_1. For the payoffs, if a target i is the attacker's choice and is also part of a patrol schedule, then the defender would gain a reward R_i^d while the attacker would receive a penalty P_i^a, else the defender would receive a penalty P_i^d and the attacker would gain a reward R_i^a. Furthermore, let G_{ij}^d be the payoff for the defender if the defender chooses patrol j and the attacker chooses to attack target i. G_{ij}^d can be represented as a linear combination of the defender reward/penalty on target i and A_{ij}, the effectiveness probability of the defensive activity performed on target i for patrol j, as described by Equation 1. The value of A_{ij} is 0 if target i is not in patrol j.

$$G_{ij}^d = A_{ij}R_i^d + (1 - A_{ij})P_i^d \quad (1)$$

For instance, suppose target 1 is covered using k_1 in strategy 5, and the value of A_{15} is 0.5. If $R_1^d = 150$ and $P_1^d = -50$, then $G_{15}^d = 0.5(150) + (1 - 0.5)(-50) = 50$. ($G_{ij}^a$ would be computed in a similar fashion.) If a target is visited multiple times with different activities, only the highest quality activity is considered.

In the USCG problem, rewards and penalties are based on an analysis completed by a contracted company of risk analysts that looked at the targets in the port of Boston and assigned corresponding values for each one. The types of factors taken into consideration for generating these values include economic damage and injury/loss of life. Meanwhile, the effectiveness probability, A_{ij}, for different defensive activities are decided based on the duration of the activities. Longer activities lead to a higher possibility of capturing the attackers. While Table 1 shows a zero-sum game, the algorithm used by PROTECT is *not limited to a zero-sum game*; the actual payoff values are determined by the USCG.

Patrol Schedule	Target 1	Target 2	Target 3	Target 4
(1:k_1), (2:k_1), (1:k_1)	50,-50	30,-30	15,-15	-20,20
(1:k_2), (2:k_1), (1:k_1)	100,-100	60,-60	15,-15	-20,20
(1:k_1), (2:k_1), (1:k_2)	100,-100	60,-60	15,-15	-20,20
(1:k_2), (2:k_1), (1:k_2)	100,-100	60,-60	15,-15	-20,20
(1:k_1), (3:k_1), (2:k_1), (1:k_1)	50,-50	30,-30	15,-15	10,-10
(1:k_1), (2:k_1), (3:k_1), (1:k_1)	50,-50	30,-30	15,-15	10,-10

Table 1: Portion of a simplified example of a game matrix

3.2 Compact Representation

In our game, the number of defender strategies, i.e. patrol schedules, grows combinatorially, generating a scale-up challenge. To achieve scale-up, PROTECT uses a compact representation of the patrol schedules using two ideas: (i) combining equivalent patrol schedules and; (ii) removal of dominated patrol schedules.

With respect to equivalence, different permutations of patrol schedules provide identical payoff results. Furthermore, if an area is visited multiple times with different activities in a schedule, only the activity that provides the defender the highest payoff requires attention. Therefore, many patrol schedules are equivalent if the set of patrol areas visited and defensive activities in the schedules are the same even if their order differs. Such equivalent patrol schedules are combined into a single compact defender strategy, represented as a set of patrol areas and defensive activities (and minus any ordering information). Table 2 presents a compact version of Table 1, which shows how the game matrix is simplified by using equivalence to form compact defender strategies, e.g. the patrol schedules in the rows 2-4 from Table 1 are represented as a compact strategy Γ_2 = {(1,k_2), (2,k_1)} in Table 2.

Compact Strategy	Target 1	Target 2	Target 3	Target 4
Γ_1 = {(1:k_1), (2:k_1)}	50,-50	30,-30	15,-15	-20,20
Γ_2 = {(1:k_2), (2:k_1)}	100,-100	60,-60	15,-15	-20,20
Γ_3 = {(1:k_1), (2:k_1), (3:k_1)}	50,-50	30,-30	15,-15	10,-10

Table 2: Example compact strategies and game matrix

Next, the idea of dominance is illustrated using Table 2 and noting the difference between Γ_1 and Γ_2 is the defensive activity on patrol area 1. Since activity k_2 gives the defender a higher payoff than k_1, Γ_1 can be removed from the set of defender strategies because Γ_2 covers the same patrol areas while giving a higher

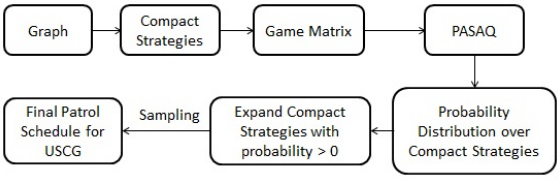


Figure 2: Flow chart of the PROTECT system

payoff for patrol area 1. To generate the set of compact defender strategies, a naive approach would be to first generate the full set of patrol schedules and then prune the dominated and equivalent schedules. Instead, PROTECT uses three ideas to quickly compute the compact strategies: (i) computation of a starting point for compact strategy generation; (ii) computation of a stopping point and; (iii) verification of feasibility in compact strategies.

While generating compact strategies, we first generate compact strategies containing $\hat{n}$ patrol areas, then $\hat{n}-1$ patrol areas and so on until $\check{n}$ patrol areas. $\hat{n}$ is called the starting point and is defined as τ/ρ where τ is the maximum patrol time and ρ shortest duration of a defensive activity. The maximum number of areas in any compact strategy must be less than or equal to $\hat{n}$. For example, if there are 20 patrol areas, τ =100 minutes and $\rho = 10$ minutes, then the algorithm will start by generating compact strategies with 10 patrol areas. It must be verified that a feasible patrol schedule can be formed from each compact strategy. This is achieved by constructing the shortest patrol schedule that is equivalent to the compact strategy, and comparing the patrol travel time against τ.

Let $S(n)$ represent all the compact strategies that contain n patrol areas. If $S(\check{n})$ contains all the compact strategies that are covered with the highest quality defensive activity at each patrol area, the process of generating compact strategies will terminate and $\check{n}$ is called the stopping point of enumeration. Any compact strategy that contains fewer than $\check{n}$ patrol areas will be dominated by a compact strategy in $S(\check{n})$.

Figure 2 shows a high level view of the steps of the algorithm using the compact representation. The compact strategies are used instead of full patrol schedules to generate the game matrix. Once the optimal probability distribution is calculated (as explained in Section 3.3) for the compact strategies, the strategies with a probability greater than 0 are expanded to a complete set of patrol schedules.

In this expansion from a compact strategy to a full set of patrol schedules, we need to determine the probability of choosing each patrol schedule, since a compact strategy may correspond to multiple patrol schedules. The focus here is to increase the difficulty for the attacker to conduct surveillance by increasing unpredictability[1], which we achieve by randomizing uniformly over all expansions of the compact defender strategies. The uniform distribution provides the maximum entropy (greatest unpredictability). Thus, all the patrol schedules generated from a single compact strategy are assigned a probability of v_i/w_i where v_i is the probability of choosing a compact strategy Γ_i and w_i is the total number of expanded patrol schedules for Γ_i. The complete set of patrol schedules and the associated probabilities are then sampled and provided to the USCG, along with the start time of the patrol generated via uniform random sampling.

3.3 Human Adversary Modeling

[1]Creating optimal Stackelberg defender strategies that increase the attacker's difficulty of surveillance is an open research issue in the literature; here we choose to maximize unpredictability as the first step.

t_i	Target i
R_i^d	Defender reward on covering t_i if it's attacked
P_i^d	Defender penalty on not covering t_i if it's attack
R_i^a	Attacker reward on attacking t_i if it's not covered
P_i^a	Attacker penalty on attacking t_i if it's covered
A_{ij}	Effectiveness probability of compact strategy Γ_j on t_i
a_j	Probability of choosing compact strategy Γ_j
J	Total number of compact strategies
x_i	Marginal coverage on t_i

Table 3: PASAQ notation as applied to PROTECT

While previous game-theoretic security applications have assumed a perfectly rational attacker, PROTECT takes a step forward by addressing this limitation of classical game theory. Instead, PROTECT uses a model of a boundedly rational adversary by using a quantal response (QR) model of an adversary, which has shown to be a promising model of human decision making [10, 11, 15]. A recent study demonstrated the use of QR as an effective prediction model of humans [14]. An even more relevant study of the QR model was conducted by Yang et al. [15] in the context of security games where this model was shown to outperform competitors in modeling human subjects. Based on this evidence, PROTECT uses a QR model of a human adversary. (Aided by a software assistant, the defender still computes the optimal mixed strategy.)

The QR model adapts ideas from the literature which presumes that humans will choose better actions at a higher frequency, but with noise added to the decision making process following a logit distribution as defined below

$$q_i = \frac{e^{\lambda G_i^a(x_i)}}{\sum_{j=1}^{T} e^{\lambda G_j^a(x_i)}} \tag{2}$$

The parameter λ represents the amount of noise in the attacker's strategy. λ can range from 0 to ∞ with a value of 0 representing a uniform random probability over attacker strategies while a value of ∞ representing a perfectly rational attacker. q_i corresponds to the probability that the attacker chooses a target i; $G_i^a(x_i)$ corresponds to the attacker's expected utility of attacking target i given x_i, the probability that the defender covers target i; and T is the total number of targets.

To apply the QR model in a Stackelberg framework, PROTECT employs an algorithm known as PASAQ [16]. PASAQ computes the optimal defender strategy (within a guaranteed error bound) given a QR model of the adversary by solving the following non-linear and non-convex optimization problem P, with Table 3 listing the notation:

$$\mathtt{P}: \begin{cases} \max\limits_{x,a} \dfrac{\sum_{i=1}^{T} e^{\lambda R_i^a} e^{-\lambda(R_i^a - P_i^a)x_i}((R_i^d - P_i^d)x_i + P_i^d)}{\sum_{i=1}^{T} e^{\lambda R_i^a} e^{-\lambda(R_i^a - P_i^a)x_i}} \\ x_i = \sum\limits_{j=1}^{J} a_j A_{ij}, \quad \forall i \\ \sum\limits_{j=1}^{J} a_j = 1 \\ 0 \le a_j \le 1, \quad \forall j \end{cases}$$

The first line of the problem corresponds to the computation of the defender's expected utility resulting from a combination of Equations 1 and 2. Unlike previous applications [8, 12], x_i in this case not just summarizes presence or absence on a target, but also the effectiveness probability A_{ij} on the target as well.

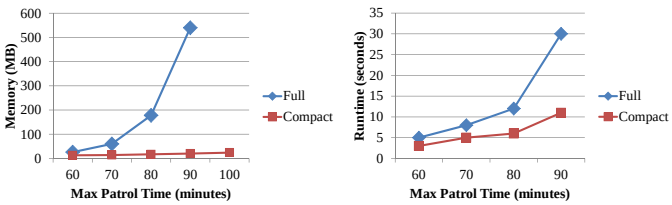
As with all QR models, a value for λ is needed to represent the noise in the attacker's strategy. Based on discussions with USCG experts about the attacker's behavior, a λ value of 0 (uniform random) and ∞ (fully rational) were ruled out. Given the payoff data for Boston, an attacker's strategy with $\lambda = 4$ starts approaching a fully rational attacker — the probability of attack focuses on a single target. It was determined from the knowledge gathered from USCG that the attacker's strategy is best modeled with a λ value that is in the range [0.5, 4]. A discrete sampling approach was used to determine a λ value that gives the highest average expected utility across attacker strategies within this range to get $\lambda = 1.5$. Selecting an appropriate value for λ remains a complex issue however, and it is a key agenda item for future work.

4. EVALUATION

This section presents evaluations based on (i) experiments completed via simulations and (ii) real-world patrol data along with USCG analysis. All scenarios and experiments, including the payoff values and graph (composed of 9 patrol areas), were based off the port of Boston. The defender's payoff values have a range of [-10,5] while the attacker's payoff values have a range of [-5,10]. The game was modeled as a zero-sum game[2] in which the attacker's loss or gain is balanced precisely by the defender's gain or loss. For PASAQ, the defender's strategy uses $\lambda = 1.5$ as mentioned in Section 3.3. All experiments are run on a machine with an Intel Dual Core 1.4 GHz processor and 2 GB of RAM.

4.1 Memory and Run-time Analysis

This section presents the results based on simulation to show the efficiency in memory and run-time of the compact representation versus the full representation (Section 3.2). In Figure 3(a), the x-axis is the maximum patrol time allowed and the y-axis is the memory needed to run PROTECT. In Figure 3(b), the x-axis is the maximum patrol time allowed and the y-axis is the run-time of PROTECT. The maximum patrol time allowed determines the number of combinations of patrol areas that can be visited — so the x-axis indicates a scale-up in the number of defender strategies. When the maximum patrol time is set to 90 minutes, the full representation takes 30 seconds and uses 540 MB of memory while the compact representation takes 11 seconds to run and requires 20 MB of memory. Due to the exponential increase in the memory and run-time that is needed for the full representation, it cannot be scaled up beyond 90 minutes.



(a) Memory comparison (b) Runtime comparison

Figure 3: Comparison of full vs. compact representation

4.2 Utility Analysis

Given that we are working with real data, it is useful to understand whether PROTECT using PASAQ with $\lambda = 1.5$ provides an advantage when compared to: (i) a uniform random defender's strategy; (ii) a mixed strategy with the assumption of the attacker attacking any target uniformly at random ($\lambda = 0$) or; (iii) a mixed strategy assuming a fully rational attacker ($\lambda = \infty$). The previously existing DOBSS algorithm was used for $\lambda = \infty$ [12]. Additionally, comparison with the $\lambda = \infty$ approach is important because of the extensive use of this assumption in previous applications (for our zero-sum case, DOBSS is equivalent to minimax but the utility does not change). Typically, we may not have an estimate of the exact value of the attacker's λ value, only a possible range. Therefore, ideally we would wish to show that PROTECT (with $\lambda = 1.5$) provides an advantage over a range of λ values assumed for the attacker (not just over a point estimate), justifying our use of the PASAQ algorithm.

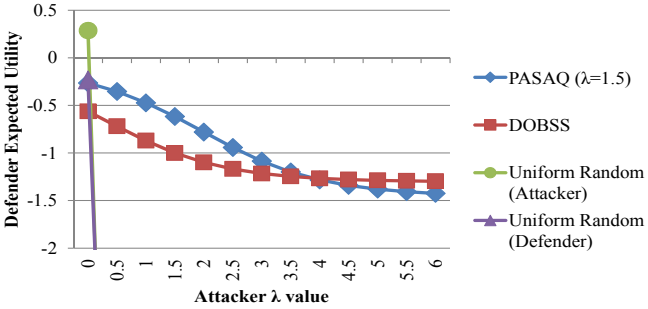


Figure 4: Defender's Expected Utility when varying λ for attacker's strategy(color)

To achieve this, we compute the average defender utility of the four approaches above as the λ value of the attacker's strategy changes from [0, 6], which subsumes the range [0.5, 4] of reasonable attacker strategies. In Figure 4, the y-axis represents the defender's expected utility and the x-axis is the λ value that is used for the attacker's strategy. Both uniform random strategies perform well when the attacker's strategy is based on $\lambda = 0$. However, as λ increases, both strategies quickly drop to a very low defender expected utility. In contrast, the PASAQ strategy with $\lambda = 1.5$ provides a higher expected utility than that assuming a fully rational attacker over a range of attacker λ values (and indeed over the range of interest), not just at $\lambda = 1.5$.

4.3 Robustness Analysis

In the real world, observation, execution, and payoffs, are not always perfect due to the following: noise in the attacker's surveillance of the defender's patrols, the many tasks and responsibilities of the USCG where the crew may be pulled off a patrol, and limited knowledge of the attacker's payoff values. Our hypothesis is that PASAQ with $\lambda = 1.5$ is more robust to such noise than a defender strategy which assumes full rationality of the attacker such as DOBSS [12], i.e. PASAQ's expected defender utility will not degrade as much as DOBSS over the range of attacker λ of interest. This is illustrated by comparing both PASAQ and DOBSS against observation, execution, and payoff noise [8, 9, 17]. (A comparison of the uniform random strategies was not included due to its poor performance shown in Figure 4.) All experiments were run generating 200 samples with added noise and averaging over all the samples. For Figures 5, 6, and 7, the y-axis represents the defender's expected utility and the x-axis is the attacker's λ value, with error bars depicting the standard error.

The first experiment considers observational noise, which means that the attacker has noise associated with observing the defender's patrol strategy as shown in Figure 5. In this scenario, if the defender

[2]In general these types of security games are non-zero-sum [12], however for Boston as a first step it was decided to cast the game as zero-sum.

covered a target with probability p, the attacker may perceive the probability to be uniformly distributed in $[p - x, p + x]$ where x is the noise. The low observation error corresponds to $x = 0.1$ while for high error $x = 0.2$. Contrary to expectation, observation error leads to an increase in defender expected utility in PASAQ, but a potential decrease (or no change) in DOBSS — thus PASAQ ends up dominating DOBSS by a larger margin over bigger ranges of λ, further consolidating the reason to use PASAQ rather than a full-rationality model.

An example illustrates PASAQ's unexpected behavior. Suppose the defender's strategy is $\mathbf{c}$ and there are two targets, t_1 and t_2 with defender expected utilities of $U_1^d(\mathbf{c}) = -2$ and $U_2^d(\mathbf{c}) = -1$, with the attacker's expected utility $U^a(\mathbf{c})$ being the opposite because this is a zero-sum game. For an attacker strategy with a higher λ, the adversary will choose to attack t_1 and the defender would get a utility of -2. When observation noise is added, increases in the coverage of t_1 results in decreases in $U_1^a(\mathbf{c}')$ so the attacker might choose to attack t_2 instead, giving the defender a higher utility than when noise is absent. If the coverage of t_1 decreases, $U_1^a(\mathbf{c}')$ will increase and the attacker will still choose to attack t_1, but $U_1^d(\mathbf{c})$ will remain the same as when there was no noise.

The reason there is a different trend for DOBSS is because DOBSS minimizes the maximum attacker's expected utility or, in our situation, also maximizes the minimum defender's expected utility. This results in multiple targets with the same minimum defender's utility; these targets are referred to as an *attack set* [12]. Typically, when the coverage over the attack set varies due to observation error, some of the targets have less and some have more coverage, but the attacker ends up attacking the targets in the attack set regardless, giving the defender almost no change in its expected utility.

For the second experiment, noise is added to the execution phase of the defender as shown in Figure 6. If the defender covered a target with probability p, this probability now changes to be uniformly distributed in $[p - x, p + x]$ where x is the noise. The low execution error corresponds to $x = 0.1$ whereas high error corresponds to $x = 0.2$. The key takeaway here is that execution error leads to PASAQ dominating DOBSS over all tested values of λ, further strengthening the reason to use PASAQ rather than a full-rationality model. When execution error is added, PASAQ dominates DOBSS because the latter seeks to maximize the minimum defender's expected utility so multiple targets will have the same minimum defender utility. For DOBSS, when execution error is added, there is a greater probability that one of these targets will have less coverage, resulting in a lower defender's expected utility. For PASAQ, typically only one target has the minimum defender expected utility. As a result changes in coverage do not impact it as much as DOBSS. Similar to observation error, as execution error increases, the advantage in the defender's expected utility of PASAQ over DOBSS increases even more.

In the third experiment shown in Figure 7, payoff noise is added by aggregating mean-0 Gaussian noise to the attacker's original payoff values (similar to [8]). As more noise is added to the payoffs, both defenders' strategies result in an increase in the defender's expected utility because the game is no longer zero-sum. The low payoff noise corresponds to a standard deviation of 1 while a high payoff noise corresponds to a standard deviation of 1.5. Similar to the previous experiments, when payoff noise is added, DOBSS is dominated by PASAQ, indicating the robustness of PASAQ. As noise is added to the attacker's payoff but not the defender's payoff, the attacker's strategy may no longer result in the lowest possible defender expected utility. For example, with no payoff noise, target t_1 gives the attacker the highest utility and the defender the lowest utility. When noise is added to the attacker's payoffs, t_1 may no longer give the attacker the highest utility; instead, he/she will choose to attack target t_2, and the defender receives a higher utility than t_1. In essence, with a zero-sum game, the defender has planned a conservative strategy, based on maximin, and as such any change in the attacker is to the defender's benefit in this case.

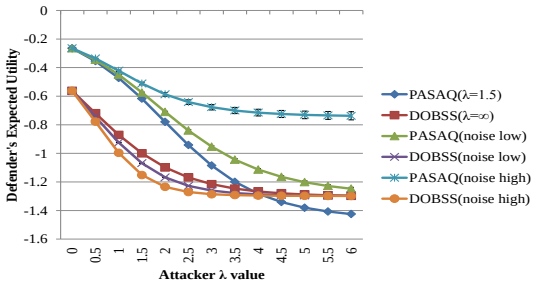


Figure 5: Defender's expected utility: Observation noise(color)

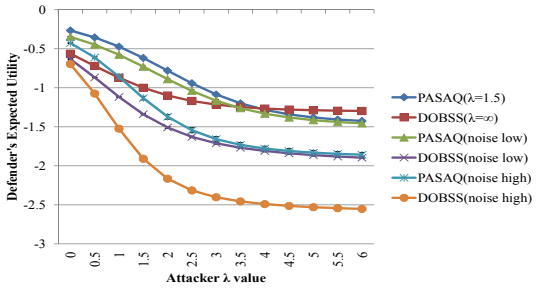


Figure 6: Defender's expected utility: Execution noise(color)

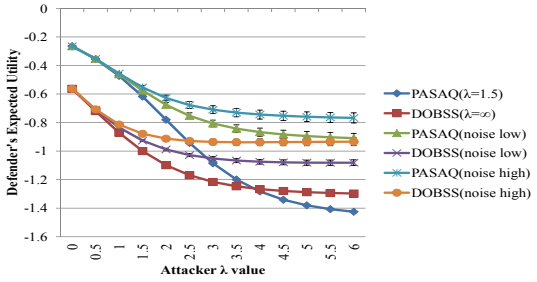


Figure 7: Defender's expected utility: Payoff noise(color)

4.4 USCG Real-World Evaluation

In addition to the data made available from simulations, the USCG conducted its own real-world evaluation of PROTECT. With permission, some aspects of the evaluation are presented in this paper.

Real-world scheduling data: Unlike prior publications at AAMAS of real-world applications of game theory for security, a key novelty of this paper is the inclusion of actual data from USCG patrols before and after the deployment of PROTECT at the port of Boston. Figure 8 and Figure 9 show the frequency of visits by USCG to different patrol areas over a number of weeks. The x-axis is the day of the week, and the y-axis is the number of times a patrol area is visited for a given day of the week. The y-axis is intentionally blurred for security reasons as this is real data from Boston. There are more lines in Figure 8 than in Figure 9 because during the implementation of PROTECT, new patrol areas were formed which contained more targets and thus fewer patrol areas in the post-PROTECT figure. Figure 8 depicts a definite pattern in the

patrols. While there is a spike in patrols executed on Day 5, there is a dearth of patrols on Day 2. Besides this pattern, the lines in Figure 8 intersect, indicating that some days, a higher value target was visited more often while on other days it was visited less often, even though the value of a target does not change day-to-day. This means that there was not a consistently high frequency of coverage of higher value targets before PROTECT.

In Figure 9, we notice that the pattern of low patrols on Day 2 (from Figure 8) disappears. Furthermore, lines do not frequently intersect, i.e. higher valued targets are visited consistently across the week. The top line in Figure 9 is the base patrol area and is visited at a higher rate than all other patrol areas.

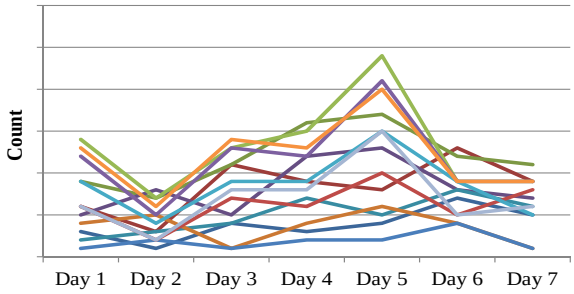


Figure 8: Patrol visits per day by area - pre-PROTECT(Color)

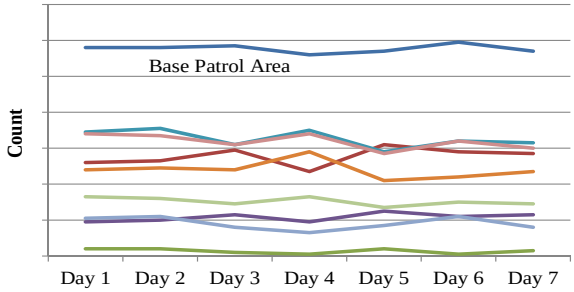


Figure 9: Patrol visits per day by area - post-PROTECT(Color)

Adversary Perspective Teams(APT): To obtain a better understanding of how the adversary views the potential targets in the port, the USCG created the Adversarial Perspective Team (APT), a mock attacker team. The APT provides assessments from the terrorist perspective and as a secondary function, assesses the effectiveness of the patrol activities before and after deployment of PROTECT. In their evaluation, the APT incorporates the adversary's known intent, capabilities, skills, commitment, resources, and cultural influences. In addition, it screens attack possibilities and assists in identifying the level of deterrence projected at and perceived by the adversary. For the purposes of this research, the adversary is defined as an individual(s) with ties to al-Qa'ida or its affiliates.

The APT conducted a pre- and post-PROTECT assessment of the system's impact on an adversary's deterrence at the port of Boston. This analysis uncovered a positive trend where the effectiveness of deterrence increased from the pre- to post- PROTECT observations.

Additional Real-world Indicators: The use of PROTECT and APT's improved guidance given to boat crews on how to conduct the patrol jointly provided a noticeable increase in the quality and effectiveness of the patrols. Prior to implementing PROTECT, there were no documented reports of illicit activity. After implementation, USCG crews, reported more illicit activities within the port and provided a noticeable "on the water" presence with industry port partners commenting, "the Coast Guard seems to be everywhere, all the time." With no actual increase in the number of resources applied, and therefore no increase in capital or operating costs, these outcomes support the practical application of game theory in the maritime security environment.

4.5 Outcomes after Boston Implementation

After evaluating the performance and impact of PROTECT at Boston, the USCG viewed this system as a success. As a result, PROTECT is now getting deployed in the port of New York. We were presented an award for the work on the PROTECT system for the Boston Harbor which reflects USCG's recognition of the impact and value of PROTECT.

5. LESSONS LEARNED: PUTTING THEORY INTO PRACTICE

Developing the PROTECT model was a collaborative effort involving university researchers and USCG personnel representing decision makers, planners and operators. Building on the lessons reported in [12] for working with security organizations, we informed the USCG of (i) the assumptions underlying the game-theoretic approaches, e.g. full adversary rationality, and strengths and limitations of different algorithms — rather than pre-selecting a simple heuristic approach; (ii) the need to define and collect correct inputs for model development and; (iii) a fundamental understanding of how the inputs affect the results. We gained three new insights involving real-world applied research; (i) unforeseen positive benefits because security agencies were compelled to reexamine their assumptions; (ii) requirement to work with multiple teams in a security organization at multiple levels of their hierarchy and; (iii) need to prepare answers to end-user practical questions not always directly related to the "meaty" research problems.

The first insight came about when USCG was compelled to reassess their operational assumptions as a result of working through the research problem. A positive result of this reexamination prompted USCG to develop new PWCS mission tactics, techniques and procedures. Through the iterative development process, USCG reassessed the reasons why boat crews performed certain activities and whether they were sufficient. For example, instead of "covered" vs "not covered" as the only two possibilities at a patrol point, there are now multiple sets of activities at each patrol point.

The second insight is that applied research requires the research team to collaborate with planners and operators on the multiple levels of a security organization to ensure the model accounts for all aspects of a complex real world environment. Initially when we started working on PROTECT, the focus was on patrolling each individual target. This appeared to micromanage the activities of boat crews, and it was through their input that individual targets were grouped into patrol areas associated with a PWCS patrol. On the other hand, input from USCG headquarters and the APT mentioned earlier, led to other changes in PROTECT, e.g. departing from a fully rational model of an adversary to a QR model.

The third insight is the need to develop answers to end-user questions which are not always related to the "meaty" research question but are related to the larger knowledge domain on which the research depends. One example of the need to explain results involved the user citing that one patrol area was being repeated and hence, randomization did not seem to occur. After assessing this concern, we determined that the cause for the repeated visits to a patrol area was its high reward — order of magnitude greater than the rarely visited patrol areas. PROTECT correctly assigned patrol schedules that covered the more "important" patrol areas more frequently. In another example, the user noted that PROTECT did not

assign any patrols to start at 4:00 AM or 4:00 PM over a 60 day test period. They expected patrols would be scheduled to start at any hour of the day, leading them to ask if there was a problem with the program. This required us to develop a layman's briefing on probabilities, randomness, and sampling. With 60 patrol schedules, a few start hours may not be chosen given our uniform random sampling of the start time. These practitioner-based issues demonstrate the need for researchers to not only be conversant in the algorithms and math behind the research, but also be able to explain from a user's perspective how solutions are accurate. An inability to address these issues would result in a lack of real-world user confidence in the model.

6. SUMMARY AND RELATED WORK

This paper reports on PROTECT, a game-theoretic system deployed by the USCG in the port of Boston since April 2011 for scheduling their patrols. USCG has deemed the deployment of PROTECT in Boston a success and efforts are underway to deploy PROTECT in the port of New York, and to other ports in the United States. PROTECT uses an attacker-defender Stackelberg game model, and includes five key innovations.

First, PROTECT moves away from the assumption of perfect adversary rationality seen in previous work, relying instead on a quantal response (QR) model of the adversary's behavior. While the QR model has been extensively studied in the realm of behavioral game theory, to the best of our knowledge, this is its first real-world deployment. Second, to improve PROTECT's efficiency, we generate a novel compact representation of the defender's strategy space, exploiting equivalence and dominance. Third, the paper shows how to practically model a real-world (maritime) patrolling problem as a Stackelberg game. Fourth, we provide experimental results illustrating that PROTECT's QR model of the adversary is better able to handle real-world uncertainties than a perfect rationality model. Finally, for the first time in a security application evaluation, we use real-world data: (i) providing a comparison of human-generated security schedules versus those generated via a game-theoretic algorithm and; (ii) results from an APT's analysis of the impact of the PROTECT system. The paper also outlined the insights from the project which include the ancillary benefits due to a review of assumptions made by security agencies, and the need for knowledge to answer questions not directly related to the research problem.

As a result, PROTECT has advanced the state of the art beyond previous applications of game theory for security. Prior applications mentioned earlier, including ARMOR, IRIS or GUARDS [12], have each provided unique contributions in applying novel game-theoretic algorithms and techniques. Interestingly, these applications have revolved around airport and air-transportation security. PROTECT's novelty is not only its application domain in maritime patrolling, but also in the five key innovations mentioned above, particularly its emphasis on moving away from the assumption of perfect rationality by using the QR model.

In addition to game-theoretic applications, the issue of patrolling has received significant attention in the multi-agent literature. These include patrol work done by robots primarily for perimeter patrols that have been addressed in arbitrary topologies [2], maritime patrols in simulations for deterring pirate attacks [13], and in research looking at the impact of uncertainty in adversarial behavior [1]. PROTECT differs from these approaches in its use of a QR model of a human adversary in a game theoretic setting, and in being a deployed application. Building on this initial success of PROTECT, we hope to deploy it at more and much larger-sized ports. In so doing, in the future, we will consider significantly more complex attacker strategies, including potential real-time surveillance and coordinated attacks.

7. ACKNOWLEDGMENTS

We thank the USCG offices, and particularly sector Boston, for their exceptional collaboration. The views expressed herein are those of the author(s) and are not to be construed as official or reflecting the views of the Commandant or of the U.S. Coast Guard. 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 award number 2010-ST-061-RE0001.

8. REFERENCES

[1] N. Agmon, S. Kraus, G. A. Kaminka, and V. Sadov. Adversarial uncertainty in multi-robot patrol. In *IJCAI*, 2009.

[2] N. Basilico, N. Gatti, and F. Amigoni. Leader-follower strategies for robotic patrolling in environments with arbitrary topologies. In *AAMAS*, 2009.

[3] D. Blair. Annual threat assessment of the US intelligence community for the senate select committee on intelligence. *http://www.dni.gov/testimonies/20100202_testimony.pdf*, 2010.

[4] C. F. Camerer. *Behavioral Game Theory: Experiments in Strategic Interaction*. Princeton University Press, 2003.

[5] V. Conitzer and T. Sandholm. Computing the optimal strategy to commit to. In *ACM EC*, 2006.

[6] K. Dozier. Bin laden trove of documents sharpen US aim. *http://www.msnbc.msn.com/id/43331634/ns/us_news-security/t/bin-laden-trove-documents-sharpen-us-aim/*, 2011.

[7] D. Fudenberg and J. Tirole. *Game Theory*. MIT Press, 1991.

[8] C. Kiekintveld, J. Marecki, and M. Tambe. Approximation methods for infinite bayesian Stackelberg games: modeling distributional uncertainty. In *AAMAS*, 2011.

[9] D. Korzhyk, V. Conitzer, and R. Parr. Solving Stackelberg games with uncertain observability. In *AAMAS*, 2011.

[10] R. D. McKelvey and T. R. Palfrey. Quantal response equilibria for normal form games. *Games and Economic Behavior*, 10(1):6–38, 1995.

[11] B. W. Rogers, T. R. Palfrey, and C. F. Camerer. Heterogeneous quantal response equilibrium and cognitive hierarchies. *Journal of Economic Theory*, 2009.

[12] M. Tambe. *Security and Game Theory: Algorithms, Deployed Systems, Lessons Learned*. Cambridge University Press, 2011.

[13] O. Vanek, M. Jakob, O. Hrstka, and M. Pechoucek. Using multi-agent simulation to improve the security of maritime transit. In *MABS*, 2011.

[14] J. Wright and K. Leyton-Brown. Beyond equilibrium: Predicting human behavior in normal form games. In *AAAI*, 2010.

[15] R. Yang, C. Kiekintveld, F. Ordonez, M. Tambe, and R. John. Improving resource allocation strategy against human adversaries in security games. In *IJCAI*, 2011.

[16] R. Yang, M. Tambe, and F. Ordonez. Computing optimal strategy against quantal response in security games. In *AAMAS*, 2012.

[17] Z. Yin, M. Jain, M. Tambe, and F. Ordóñez. Risk-averse strategies for security games with execution and observational uncertainty. In *AAAI*, 2011.

SAVES: A Sustainable Multiagent Application to Conserve Building Energy Considering Occupants

Jun-young Kwak, Pradeep Varakantham*, Rajiv Maheswaran, Milind Tambe, Farrokh Jazizadeh, Geoffrey Kavulya, Laura Klein, Burcin Becerik-Gerber, Timothy Hayes, Wendy Wood
University of Southern California, Los Angeles, CA, 90089
*Singapore Management University, Singapore, 178902
{junyounk,maheswar,tambe,jazizade,kavulya,lauraakl,becerik,hayest,wendy.wood}@usc.edu,
*pradeepv@smu.edu.sg

ABSTRACT

This paper describes an innovative multiagent system called SAVES with the goal of conserving energy in commercial buildings. We specifically focus on an application to be deployed in an existing university building that provides several key novelties: (i) jointly performed with the university facility management team, SAVES is based on actual occupant preferences and schedules, actual energy consumption and loss data, real sensors and hand-held devices, etc.; (ii) it addresses novel scenarios that require negotiations with groups of building occupants to conserve energy; (iii) it focuses on a non-residential building, where human occupants do not have a direct financial incentive in saving energy and thus requires a different mechanism to effectively motivate occupants; and (iv) SAVES uses a novel algorithm for generating optimal MDP policies that explicitly consider multiple criteria optimization (energy and personal comfort) as well as uncertainty over occupant preferences when negotiating energy reduction – this combination of challenges has not been considered in previous MDP algorithms. In a validated simulation testbed, we show that SAVES substantially reduces the overall energy consumption compared to the existing control method while achieving comparable average satisfaction levels for occupants. As a real-world test, we provide results of a trial study where SAVES is shown to lead occupants to conserve energy in real buildings.

Categories and Subject Descriptors

I.2.11 [**ARTIFICIAL INTELLIGENCE**]: Distributed Artificial Intelligence

General Terms

Algorithms, Experimentation, Human Factors

Keywords

Innovative Applications, Energy, Sustainable Multiagent Building Application, Multi-objective Optimization

1. INTRODUCTION

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

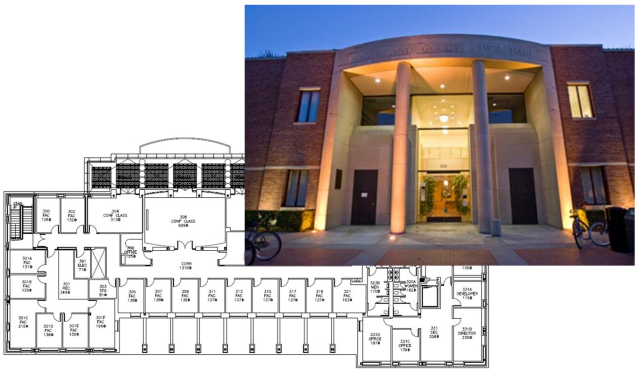


Figure 1: The actual research testbed (RGL) at the University of Southern California

Limited availability of energy sources has led to the need to develop efficient measures of conserving energy. Motivated by this need, researchers at AAMAS have been developing multiagent systems to conserve energy, both for deployment in smart grids and in buildings, with a particular focus on residential buildings [10, 16, 17, 21].

Inspired by this prior work, we describe an innovative multiagent system called SAVES (Sustainable multi-Agent building application for optimizing Various objectives including Energy and Satisfaction), where agents communicate and negotiate with human occupants to conserve energy. SAVES focuses on energy conservation in commercial (including office and educational) buildings given their significant burden on energy consumption, e.g., in 2008 buildings in the U.S. consumed 18.5 QBtu, representing 46.2% of building energy consumption and 18.4% of U.S. energy consumption [1]. To this end, this paper specifically focuses on an application to be deployed at Ralph & Goldy Lewis Hall (RGL) at the University of Southern California (shown in Figure 1).

SAVES provides the following key novelties. First, jointly performed with the university facility management team, our research is based on actual occupant preferences and schedules, actual energy consumption and loss data, real sensors and hand-held devices, etc. Second, SAVES addresses novel scenarios that require agents to negotiate with groups of building occupants to conserve energy; previous work has typically focused on agents' negotiation with individual occupants [3, 12]. Third, it focuses on non-residential buildings, where human occupants do not have a direct financial incentive in saving energy. Furthermore, commercial buildings offer new opportunities for energy conservation, since oc-

cupants may follow a more regular schedule, allowing SAVES to plan ahead for energy conservation. Finally, SAVES uses a novel algorithm for generating optimal *BM-MDP* policies that explicitly considers multiple criteria optimization (energy and personal comfort) as well as uncertainty over occupant preferences when negotiating for energy reduction – this combination of challenges has not been considered in previous MDP algorithms [5, 6, 8, 13].

We provide three sets of evaluations of SAVES. First, we constructed a detailed simulation testbed, with details all the way down to individual electrical outlets in our targeted building and variations in solar gain per day; and then validated this simulation. Within this simulation testbed, we show that SAVES substantially reduces the overall energy consumption compared to existing control methods while achieving comparable satisfaction level of occupants. Second, we show the benefits of BM-MDPs by showing that it gives a well-balanced solution while considering multiple criteria. Third, as a real-world test, we provide results of a human subject study where SAVES is shown to lead human occupants to significantly reduce their energy consumption in real buildings.

In Section 2, we describe our testbeds. This includes both the real educational building where SAVES is to be deployed, and our simulation testbed, which we validate by comparing with real building data. Next, in Section 3, we describe the SAVES multiagent system, and the novel BM-MDP algorithm at the heart of SAVES. Section 4 provides evaluations discussed above.

2. TESTBEDS

2.1 Educational Building Testbed

SAVES is to be deployed in an actual educational building. Figure 1 shows the real testbed building (RGL) and the floor plan of 3^{rd} floor. It is a multi-functional building that has been designed with a building management system, and it provides a good environment to test various control strategies to mitigate energy consumption. In particular, this campus building has three floors in total and is composed of different types of spaces including classrooms, offices for faculty and staff, and conference rooms for meetings. Each floor has a large number of rooms and zones (a set of rooms that is controlled by specific piece of equipment) with various physical properties including different building devices, orientation, window size, room size and lighting specifications. For instance, the 3^{rd} floor has 24 zones and 39 rooms.

Within this building, components and equipment include HVAC (Heating, Ventilating, and Air Conditioning) systems, lighting systems, office electronic devices such as computers and AV equipment, and different types of sensors and energy meters. Human occupants of the building are divided into two main categories: permanent and temporary. Permanent occupants include office users such as faculty, staff, researchers and laboratory residents. Temporary occupants include scheduled occupants like students or faculty attending classes or meetings and unscheduled occupants who are students or faculty using common lounges or dining spaces.

In this domain, there are two types of energy-related occupant behaviors that SAVES can influence to conserve energy use: individual behaviors and group behaviors. Individual behaviors only affect an environment where the individual is located. They include adjusting light sources and temperature in individual offices and turning on/off computers and other electronics. Group behaviors lead to changes in shared spaces and require negotiation with a group of occupants in the building. For instance, SAVES may negotiate with a group of occupants to adjust the lighting level and temperature in their shared office or to relocate a meeting to a smaller office. As we will show later, energy savings by considering such group negotiations together are significant.

The desired goal in this educational building is to optimize multiple criteria, i.e., achieve maximum energy savings without trading off the comfort level of occupants. The research on this testbed building is intended to be generalized to other building types, where we can observe many different types of energy-use and the behavioral patterns of occupants in the buildings.

2.2 Simulation Testbed

As an important first step in deploying SAVES in the actual building described in the previous section, we test SAVES in a realistic simulation environment using real building data. To that end, we have constructed a simulation testbed based on the open-source project OpenSteer (http://opensteer.sourceforge.net/), which provides a 2D, OpenGL environment. It can be used for efficient statistical analysis of different control strategies in buildings before deploying the system.

Building Components: Our simulation considers three building component categories: HVAC devices, lighting devices, and appliances. The HVAC components control the temperature of the assigned zone. The lighting devices control the lighting level of the room. The appliances in our simulation are either desktop or laptop computers. These components have two possible actions: "on" and "standby". When the lighting or appliance devices are on, they consume a fixed amount of energy. We attempt to very accurately reflect the energy consumed by each of the three component categories in the simulation. The energy consumption of HVACs is calculated based on changes in air temperature and airflow speeds, and gains from natural light source and appliances in the space. To calculate the energy consumption of the lighting and appliance devices, we collected actual energy consumption data in the testbed building. For the appliances, a desktop computer spends 0.150 *kW/h* and 0.010 *kW/h* when it is on and standby, respectively. A laptop computer spends 0.050 *kW/h* when it is on and 0.005 *kW/h* when it is on standby.[1]

Human Occupants: We built two types of human occupants in our simulation using the agent behavior framework presented in [20]. Permanent occupants stay in their offices or follow their regular schedules. Temporary occupants stay in the building for classes and leave once classes end. Each occupant has access to a subset of the six available behaviors according to her/his type — wander, attend class, go to meeting, teach, study, and perform research — any one of which may be active at a given time, where the behavior is selected based on class and meeting schedules. Occupants also have a satisfaction level based on the current environment, modeled as a percentage between 0 and 100 (0 is fully dissatisfied, 100 is fully satisfied).

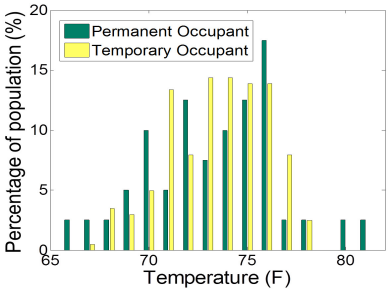


Figure 2: Actual Temp. Preference

To model the satisfaction level in this simulation, we use a Gaus-

[1]The detailed equations to compute the energy consumption and actual parameter values are presented here: http://teamcore.usc.edu/junyounk/energy/AAMAS12-SAVES-supplementary.pdf

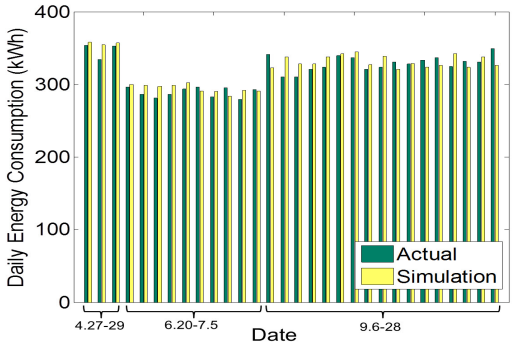


Figure 3: Energy Consumption Validation

sian distribution $N(\mu, \sigma)$ for each occupant. The mean (μ) of each individual Gaussian is drawn from actual occupant preference data shown in Figure 2 (e.g., for 18% of permanent occupants, μ=76°F). This data was gathered from 40 permanent occupants and 202 temporary occupants in RGL over two weeks in the spring of 2011. We use this actual data instead of the ASHRAE standard, which fails to account for individual preferences. The standard deviation (σ) of each Gaussian is selected uniformly randomly from a range of 3–5°F [11]. Based on the constructed Gaussian model for each occupant, the satisfaction level is computed as follows:

$$S(t) = \begin{cases} 100, & \text{if } t = \mu \\ \frac{f(t)}{f(\mu)} \times 100, & \text{if } t \neq \mu \end{cases} \quad (1)$$

where $S(t)$ is the satisfaction function, $f(x)$ is the probability density function of $N(\mu, \sigma)$, and t is the current temperature.

Validation: Before testing SAVES in simulation, we first validate the simulation testbed. Specifically, we compare the energy consumption calculated in the simulation testbed with actual energy meter data using the 3^{rd} floor of the actual testbed building.

Figure 3 shows that daily energy use comparison data (y-axis) measured for 30 sample weekdays throughout different seasons (x-axis; 3 weekdays in 2011 Spring, 10 weekdays in 2011 Summer, 17 weekdays in 2011 Fall). The energy consumption includes the amount consumed by HVACs, lighting devices and appliances. We use measured parameter values such as solar gain and outdoor temperature and real parameter values for the building obtained from the facility management system. We set the starting indoor temperature using real data. The likelihood value for human occupants to "turn off" lights and appliances when they leave their offices is 76%, based on a survey of the testbed building. Students follow 2010 Fall, 2011 Spring and 2011 Fall class schedules, and faculty, staff and students follow their meeting schedules.

As shown in the figure, the difference between actual energy meter data and energy use from the simulation testbed was between 0.17% – 8.71% (mean difference: 3.37%), which strongly supports our claim that the simulation testbed is realistic.

3. SAVES

In this section, we describe the key components of SAVES and how to optimally plan negotiations with groups of occupants to conserve energy in our application.

3.1 Agents in SAVES

At the heart of SAVES are two types of agents: room agents and proxy agents (Figure 4). There is a dedicated room agent per office and conference room, in charge of reducing energy consumption in that room. It can access sensors to retrieve information such as the

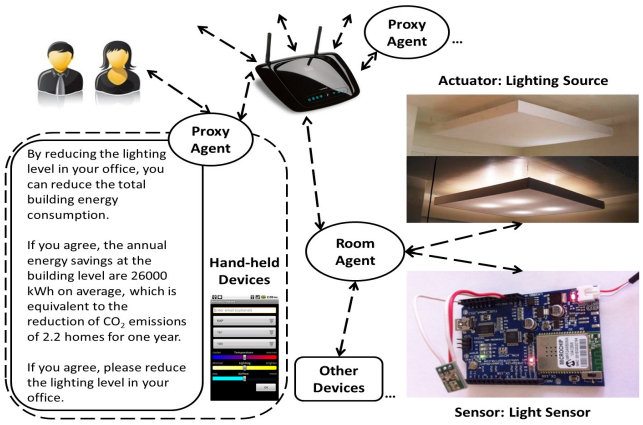


Figure 4: Agents & Communication Equipment in SAVES. An agent in SAVES sends feedback including energy use to occupants.

current lighting level and temperature and energy use at different levels (building-level, floor-level, zone-level, and room-level) and impact the operation of actuators. A proxy agent [18] is on an individual occupant's hand-held device and it has the corresponding occupant's preference and behavior models. Proxy agents communicate on behalf of an occupant to the room agent. Such proxy agents' adjustable autonomy – when to interrupt a user and when to act autonomously – is recognized as a major research issue [18, 19], but since it is not our focus, we use preset rules instead. Room agents may directly communicate with occupants without proxy agents if needed. Finally, different room agents coordinate among themselves via proxy agents, e.g., if two separate conference room agents wish to move a meeting to one occupant's office, the proxy of that occupant allows one of the room agents to proceed, blocking the other's request (see Figure 4).

Room agent reasoning is based on a new model called *Bounded parameter Multi-objective MDPs* (BM-MDPs), which is one of the contributions of this research. BM-MDPs are a hybrid of MO-MDPs [5, 13] and BMDPs [8]. BM-MDPs are responsible for planning simple and complex tasks. Simple tasks include turning on the HVAC before a class or a meeting, and do not need the full power of the BM-MDPs. Complex tasks were why BM-MDPs were created; these include negotiating with groups of individuals to relocate meetings to smaller rooms to save energy, negotiating with multiple occupants of a shared office to reduce energy usage in the form of lights or HVACs, and others. Before describing BM-MDPs in depth, we motivate their use by elaborating on the meeting relocation negotiation scenario.

Group Meeting Relocation Negotiation Example Consider a meeting that has been scheduled with two attendees (P_1 and P_2) in a large conference room that has more light sources and appliances than smaller offices. Since the meeting has few attendees, the room agent can negotiate with attendees to relocate the meeting to nearby small, sunlit offices, which can lead to significant energy savings. The room agent handles this negotiation based on BM-MDPs. There are three objectives (i.e., three separate reward functions) that the room agent needs to consider during this negotiation: i) energy saving (R_1), ii) P_1's comfort level change (R_2), and iii) P_2's comfort level change (R_3). The room agent first checks the available offices. Assuming there are two available offices A and B, the room agent asks each attendee if she or he will agree to relocate the meeting to one of the available offices. In asking an attendee, the room agent must consider the uncertainty of whether an attendee is likely to accept its offer to relocate

the meeting. Since asking incurs a cost (e.g., cost caused by interrupting people), the room agent needs to reason about which option is preferable considering P_1 and P_2's likelihood to accept each option (A or B) and the reward functions for each option to reduce the required cost and maximize benefits. Assuming A is preferable, the optimal policy of the agent is "ask P_1 first about A"–"if P_1 accepts, ask P_2 about A"–"if P_1 does not reply, ask P_1 about A again"–"repeat the process with B"–"if both agree, relocate the meeting"–"if both disagree, find other available options." While this is a simplified example, in practice the problem is more difficult, as there may be more than two attendees in a meeting. The room agent must also first communicate with the proxies of the owners of offices A and B and there may be uncertainty in their agreement to have a meeting in their office; further adding to the challenge of sequential decision making under uncertainty. In addition, the agent must decide if it should ask P_1 first and use that result to influence P_2, etc.

Thus, BM-MDPs must reason with multiple objectives, but simultaneously must reason with the uncertainty in the domain. In fact, in a complex domain such as ours, the probabilities of attendees' or others' acceptance of the room agent's offer, or the probabilities of other outcomes may not be precisely known — we may only have a reasonable upper and lower bound over such probabilities. Indeed, precisely knowing the model is very challenging, and we ended up building BM-MDPs to address both these challenges and requirements. However, before explaining BM-MDPs, we first explain MO-MDPs on which BM-MDPs are built.

3.2 Multi-objective MDPs

The negotiation scenarios described earlier require SAVES to consider multiple objectives simultaneously: energy consumption and satisfaction level of multiple individuals. To handle such multiple objectives, MDPs have been extended to take into account multiple criteria assuming no *model uncertainty*. *Multi-Objective MDPs* (MO-MDPs) [5, 13] are defined as an MDP where the reward function has been replaced by a vector of rewards. Specifically, MO-MDPs are described by a tuple $\langle S, A, T, \{R_i\}, p\rangle$, where R_i is the reward function for objective i and p denotes the starting state distribution ($p(s) \geq 0$). In the *meeting relocation example* shown in Section 3.1, specifically, the multiple reward functions, $\{R_i\}$, include energy consumption (which is the reduction in energy usage in moving from a conference room to a smaller office), and comfort level defined separately for each individual (based on data related to their temperature comfort zones).

The key takeaway from MO-MDPs towards BM-MDPs is an understanding of how to generate a policy in the presence of such multiple objectives that are not aggregated into one single value. The key principle we rely on, given the current domain of non-residential buildings is one of fairness; we wish to reduce energy usage, but we cannot sacrifice any one individual's comfort entirely in service of this goal. To meet this requirement, we focus on minimizing the maximum *regret* instead of maximizing the reward value based on a min-max optimization technique [14] to get a well-balanced solution.

To minimize the maximum regret, we first need to compute the optimal value for each objective using the MDP framework relying on the following standard formulation:

$$\min V^*(s) \tag{2}$$

$$\text{s.t.} \quad V^*(s) \geq R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \cdot V^*(s'), \tag{3}$$

$$0 \leq \gamma < 1 \tag{4}$$

where V^* is an optimal value, and γ is a discount factor.

We define the regret in MO-MDPs as following:

Definition Let $H_i^\alpha(s)$ be the *regret* with respect to a policy α for objective i and state s. Formally,

$$H_i^\alpha(s) = V_i^{\alpha_i^*}(s) - V_i^\alpha(s), \tag{5}$$

where $V_i^{\alpha_i^*}(s)$ is the value of the *optimal* policy, α_i^*, and $V_i^\alpha(s)$ is the value of the policy α for objective i and state s.

Therefore, we can minimize the maximum regret in MO-MDPs using the following optimization problem:

$$\min D \tag{6}$$

$$\text{s.t.} \quad D \geq \sum_{s \in S} p(s) \cdot \left[V_i^*(s) - V_i(s)\right], \forall i \in I, \tag{7}$$

$$V_i(s) = \sum_{a \in A} \alpha(s,a) \left[R_i(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \cdot V_i(s') \right], \tag{8}$$

$$\sum_{a \in A} \alpha(s,a) = 1, \forall s \in S, \qquad 0 \leq \gamma < 1 \tag{9}$$

where V_i^* is the constant value pre-calculated by (2) of the MDP formulation using the reward function for objective i, R_i, and I is a set of objectives.

Unfortunately, in BM-MDPs, we have an upper and lower bound on transition probabilities and rewards, and thus this optimization problem cannot be directly used. Nonetheless, it helps us understand the key difference in minimizing max regret between MO-MDPs and BM-MDPs — specifically in addressing such upper and lower bounds in BM-MDPs, we end up with different transition probabilities T_i for each objective i, as discussed below, and hence rely on a different approach to compute regret.

3.3 BM-MDPs

We now extend MO-MDPs, using ideas from BMDPs [8], to create BM-MDPs. BMDP (represented by tuple $\langle S, A, \hat{T}, \hat{R}, p\rangle$) is an extension to the standard MDP, where upper and lower bounds on transition probabilities and rewards are provided as closed real intervals. In addition to representation of uncertainty over transition probabilities and rewards, a key takeaway for BM-MDPs from BMDPs is the algorithm to generate policies. This algorithm is based on the notion of *Order-Maximizing* MDPs [8], which selects transition probabilities from the given intervals. Order-maximizing MDPs crucially take the order of states as an input – this order is ascending if we are to return a pessimistic policy (based on lower bound values), and it is descending for an optimistic policy (based on upper bound values). More specifically, using this order as an input, order-maximizing MDPs construct the transition function, and generate a policy as an output relying on value iteration. We rely on order-maximizing MDPs to generate policies in our BM-MDPs as well (but manipulate the order of states input). To provide some intuition behind the operations of the order-maximizing MDPs, we provide a simple example to show how transition values are assigned from their intervals using the given order in the following example. For more details, please refer to [8].

Example of Order Maximizing MDPs Consider a BMDP with two states: s_1 and s_2. The transition ranges are $T(s_1, a, s_1)$ = [0.5, 0.9], $T(s_1, a, s_2)$ = [0.2, 0.6]. Let us assume that the upper bound of value is $V_{ub}(s_1)$ = 3 and $V_{ub}(s_2)$ = 2 at a certain iteration of order-maximizing MDP value iteration. In BMDP, the intuition is

that for calculating the optimistic value, we require movement to s_1 as much as possible within the given range of transition probability (since it has a higher upper bound value). Therefore to create an optimistic policy, the input to the order-maximizing MDPs is to sort the states in a descending order based on the upper bounds. Given this input, the transition probabilities in the order-maximizing MDP for calculating optimistic value would be $T'(s_1, a, s_1)$ = 0.8 because $T'(s_1, a, s_2)$ should be at least 0.2, and $T'(s_1, a, s_2)$ = (1 - 0.8). Based on these transition probabilities, we obtain a new set of expected values via value iteration, generate a new descending order, and iterate until convergence.

Similar to BMDPs, the transition and reward functions in BM-MDPs have closed real intervals. Whereas BMDPs are limited to optimizing a single objective case (i.e., the BMDP model requires one unified reward function), BM-MDPs can i) optimize over multiple objectives (i.e., a vector of reward functions) with ii) different degrees of model uncertainty. Specifically, BM-MDPs are described by a tuple $\langle S, A, \hat{T}, \{\hat{R}_i\}, p\rangle$, where $\hat{R}_i$ represents the reward function for objective i.

Algorithm 1 SOLVEBMMDP()

1: **for** $i = 1 \in I$ **do**
2: $\quad\left\langle \mathbf{V}^*_{i,lb}, \mathbf{V}^*_{i,ub} \right\rangle \leftarrow$ SolveBMDP($BMDP_i$)
3: $\{\mathbf{V}^{'}_{i,lb}\} \leftarrow \infty$; $\{\mathbf{V}_{i,lb}\} \leftarrow \mathbf{0}$
4: **while** $|\{\mathbf{V}'_{i,lb}\} - \{\mathbf{V}_{i,lb}\}| > \epsilon$ **do**
5: $\quad\{\mathbf{V}_{i,lb}\} \leftarrow \{\mathbf{V}^{'}_{i,lb}\}$
6: $\quad$**for** $i = 1 \in I$ **do**
7: $\quad\quad\mathbf{O}_i \leftarrow$ SortIncreasingOrder($\{V_{i,lb}\}$)
8: $\quad\quad\mathbf{M}_i \leftarrow$ ConstructOrderMaximizingMDP($\mathbf{O}_i$);
9: $\quad\{\mathbf{V}'_{i,lb}\} \leftarrow$ SolveMOMDPPessimistic($\{\mathbf{V}_{i,lb}\}, \{\mathbf{V}^*_{i,lb}\}, \{\mathbf{M}_i\}$)
10: $\alpha_{pes} \leftarrow$ ObtainPessimisticPolicy($\{\mathbf{V}_{i,lb}\}$)
11: $\{\mathbf{V}^{'}_{i,ub}\} \leftarrow \infty$; $\{\mathbf{V}_{i,ub}\} \leftarrow \mathbf{0}$
12: **while** $|\{\mathbf{V}'_{i,ub}\} - \{\mathbf{V}_{i,ub}\}| > \epsilon$ **do**
13: $\quad\{\mathbf{V}_{i,ub}\} \leftarrow \{\mathbf{V}'_{i,ub}\}$
14: $\quad$**for** $i = 1 \in I$ **do**
15: $\quad\quad\mathbf{O}_i \leftarrow$ SortDecreasingOrder($\{V_{i,ub}\}$)
16: $\quad\quad\mathbf{M}_i \leftarrow$ ConstructOrderMaximizingMDP($\mathbf{O}_i$);
17: $\quad\{\mathbf{V}'_{i,ub}\} \leftarrow$ SolveMOMDPOptimistic($\{\mathbf{V}_{i,ub}\}, \{\mathbf{V}^*_{i,ub}\}, \{\mathbf{M}_i\}$)
18: $\alpha_{opt} \leftarrow$ ObtainOptimisticPolicy($\{\mathbf{V}_{i,ub}\}$)
19: **return** $\{\langle \alpha_{pes}, \alpha_{opt}\rangle\}$

To solve BM-MDPs, we introduce a novel algorithm that is a hybrid of BMDPs and MO-MDPs. Specifically, our algorithm marries the minimization of max regret idea from MO-MDPs with that of order maximizing MDPs to handle uncertainty over transition function and rewards. The overall flow is described in Algorithm 1. At a higher level, we have three stages: (i) computing the optimal value bounds $\left\langle \mathbf{V}^*_{i,lb}, \mathbf{V}^*_{i,ub} \right\rangle$ for each objective i using BMDPs (lines 1–2), (ii) using the MO-MDP idea to optimize multiple objectives based on a min-max formulation (lines 3–9 & 11–17), and (iii) obtaining a policy α based on the final value functions $\langle\{\mathbf{V}_{i,lb}\}, \{\mathbf{V}_{i,ub}\}\rangle$ (lines 10 & 18). The output of this algorithm is in the form of two policies (pessimistic and optimistic), and we leave it to the user to determine which one is used.

We now describe the computation of the pessimistic policy (lines 3–10). The optimistic policy (lines 11–18) is similarly computed. The pessimistic policy minimizes the maximum regret with respect to the optimal lower bound values of all objectives ($\{V^*_{i,lb}\}$) over all states; this computation is iteratively performed in line 9. For each objective i, we first get an ascending order of states using the current lower bound values $V_{i,lb}$ (line 7) to construct the order-maximizing MDP (line 8). This set of order-maximizing MDPs, $\{\mathbf{M}_i\}$, is an input to the function SolveMOMDPPessimistic() to optimize multiple objectives by directly computing regret on line 9. This computation is performed by Eq. (6) with a different transition probability function T_i in the given $\mathbf{M}_i$ instead of T. This in turn influences the sorting order of states, and the process continues until the expected values $\{V_{i,lb}\}$ converge.

4. EVALUATION OF SAVES

In this section, we provide three sets of evaluations: two sets of results tested in the simulation testbed and a set of results tested in the real-world.

4.1 Simulation: Overall Evaluation

We evaluate the performance of SAVES using both 2^{nd} and 3^{rd} floors of RGL in the simulation environment. We test BM-MDPs using a pessimistic setting and compare it with two other control heuristics discussed below.

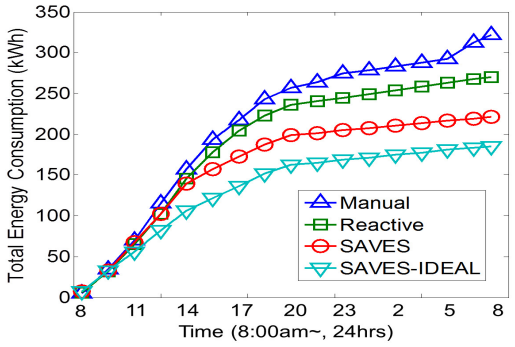
Manual Control: The manual control strategy is the baseline system that represents the current strategy operated by the facility management team in the real testbed building (RGL). In this strategy, temperature is regulated by the facility managers according to two set ranges for occupied (70°F–75°F) and unoccupied periods (50°F–90°F) of the day. In this control setting, HVACs always attempt to reach the pre-set temperature regardless of the presence of occupants and their preferences in terms of temperature. Lighting and appliance devices are controlled by human occupants. The same likelihood value for human occupants to "turn off" lights and appliances was used as in Section 2.2.

Reactive Control: We consider the reactive control heuristic for comparison purposes since it can be easily implemented using cheap sensors in the real building, and recently, some buildings have already started adopting this simple heuristic to reduce energy use. The lighting and appliance devices are now automatically controlled and turned on and off according to the presence of people. Additionally, as in [9], appropriate temperature set points of HVACs are computed based on the average preference of human occupants. HVACs automatically turn on and off according to the presence of people and temperature set points.

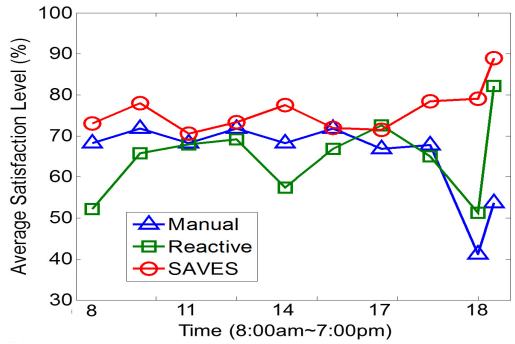
We focus on measuring two different criteria — total energy consumption (*kWh*) and average satisfaction level of occupants (%). The experiments were run on Intel Core2 Duo 2.53GHz CPU with 4GB main memory. All techniques were evaluated for 100 independent trials throughout this section. We report the average values.

4.1.1 Result: Total Energy Consumption

We compared the cumulative total energy consumption measured during 24 hours for all control strategies. Figure 5(a) shows the cumulative total energy consumption on the y-axis in *kWh* and time on the x-axis. We report the average total energy consumption measured over the same 30 weekdays used in Figure 3. As shown in the figure, the manual control strategy showed the worst result since it does not take into account behaviors or schedules of human occupants and building components simply follow the predefined policies. The reactive control strategies showed lower energy consumption than the manual setting by 16.06%. SAVES showed the best results compared to other control heuristics and statistically significant improvements (t-test; $p < 0.01$) in terms of energy used in the testbed building. Specifically, our algorithm with the ideal compliance rate (i.e., SAVES-IDEAL: occupants *always* accept the suggestions provided by the SAVES room agents to conserve energy) reduced the energy consumption by 42.45% when compared to the manual control strategy. If we use the compliance rate (68.18%) of human subjects shown in Table 3 (as measured



(a) Total Energy Consumption



(b) Average Satisfaction Level

Figure 5: Performance Evaluation of SAVES

in our real-world experiments), SAVES achieved energy savings by 31.27% (40% of the savings due to SAVES came out of group tasks, such as reducing energy consumption in shared offices, relocating meetings, and others) as compared to the manual setup. This is double the rate of the reactive approach.

4.1.2 Result: Average Satisfaction Level

Here, we compare the average satisfaction level of human occupants under different control strategies in the simulation testbed. Figure 5(b) shows the average satisfaction level in percentage on the y-axis and time on the x-axis. As shown in the results, the manual setting and our novel algorithm showed the best results. This is because the manual setting makes HVACs attempt to reach the desired temperature set point as soon as possible while disregarding the resulting energy consumption, and our method plans ahead of the schedules; thus, these two can achieve the desired comfort level faster than the reactive control strategy.

The manual strategy, however, is very sensitive to the given temperature range. In our experiment, the temperature set point was set by the facility management team (e.g., 70–75°F) based on the average preference model, thus it achieved high comfort level in the testbed. However, if the actual preferred temperature in the building is different from the average model, it fails to meet the occupant's desired level. This phenomenon can be seen when occupants stay during the unoccupied time (after typical working hours). As we can see at 18 on the x-axis (i.e., 6pm) in the figure, the average comfort level drops significantly. Due to the delayed effects in temperature change, the reactive control strategy showed significantly lower satisfaction results than other methods. For instance, it has a satisfaction level below 60% at 14 on the x-axis (i.e., 2pm). Thus, SAVES not only provides superior energy savings, but also avoids the reduction in comfort level that a reactive strategy may cause.

Table 1: Average Maximum Regret Comparison

Problem Set	MDPs	BM-MDPs	Difference
m_1	168.62	4.72	163.90
m_2	359.44	164.17	195.27
m_3	448.15	164.97	283.18
m_4	291.27	138.59	152.68
m_5	143.32	95.88	47.44

Table 2: Example of the *Meeting Relocation Negotiation*

	Max. Regret	
Objective	MDPs	BM-MDPs
Energy Savings	443.54	162.83
P_1's Comfort Lv. Change	15.34	162.84
P_2's Comfort Lv. Change	15.34	97.58

4.2 Simulation: Multi-objective Optimization

In this section, we perform more analysis on our novel algorithm. Table 1 shows the average maximum regret comparison tested in 5 different problem sets between the standard MDP with a unified reward based on the weighted sum method [22] and BM-MDPs (in this case, we assume no transition or reward uncertainty). The uniform weight distribution was applied to the weighted sum method. Our goal is to show that BM-MDPs give lower maximum regrets, which indicates well-balanced solutions as discussed earlier.

Each problem is an instance of the *meeting relocation negotiation* task, having its own reward structure but the same transition function. The problem instances are divided into five groups (problem sets m_1–m_5) based on the percentage of objectives that have positive rewards in all objectives. Recall that in the meeting relocation scenario, the different objectives include energy reduction and change in comfort level of individual participants. Specifically, in problem set m_1, relocating a meeting leads to positive rewards in over 75% of objectives (76–100%) and negative rewards in the rest of objectives, problem set m_2 has 51–75% of objectives with positive rewards, and similarly for the remaining sets, so that in problem set m_5, all objectives have negative rewards if the meeting is relocated. Each problem set has 100 independent problem instances. We then measured the average maximum regret of each method in each problem set. As shown in Table 1, BM-MDPs always showed lower maximum regrets (column 3) compared to the MDP with uniform weight (column 2), which suggests that our method gives well-balanced solutions regardless of reward characteristics.

The next question is what the well-balanced solution means in our energy domain. Let us take the *meeting relocation example* with two attendees (P_1 and P_2) discussed in Section 3.1. In Table 2, column 1 shows three objectives (energy savings and two attendees' comfort level change) and columns 2–3 indicate the maximum regret from MDPs and BM-MDPs, respectively. As shown in the table, MDPs generated a policy that almost entirely disregards energy-savings, leading to significantly large regrets (row 3, column 2). BM-MDPs, on the other hand, were able to achieve small regrets over all objectives (rows 3–5, column 3).

Lastly, we test our BM-MDP algorithm considering different degrees of model uncertainty. Figure 6 shows the average maximum regret tested over 100 different problem instances on the y-axis. We choose 1 problem from each problem set (m_1, m_2, $\cdots$, m_5) from the previous test. The noise of each model is proportional (20%) to the mean reward value and transition probability. MDPs and MO-MDPs generate policies ignoring the model's uncertainty and BM-MDPs generate two types of policies (BM-MDP-Pes: pessimistic, BM-MDP-Opt: optimistic) that explicitly account for the uncertainty. We then randomly

generate 20 different instances within the range for each problem (e.g., for m_1, we generate $m_{1,1}$, $\cdots$, $m_{1,20}$). Each generated policy is evaluated over those 20 problem instances and the average maximum regret is computed for each algorithm. For the other 4 problems (m_2, $\cdots$, m_5), we repeat the same procedure and report the overall average value. As shown in the figure, BM-MDPs have the best performance (i.e., lowest average maximum regret), which means BM-MDPs are capable of generating more robust and well-balanced solutions compared to previous work when there is model uncertainty. However, the solution quality between the pessimistic and optimistic BM-MDPs was not significantly different and their performance is domain dependent. *Note that the results shown in Figure 6 are average maximum regrets over all problem instances, and in some particular instances, MO-MDP might outperform either BM-MDP-Pes or BM-MDP-Opt (but not both even in this case).* We leave this issue for future investigation.

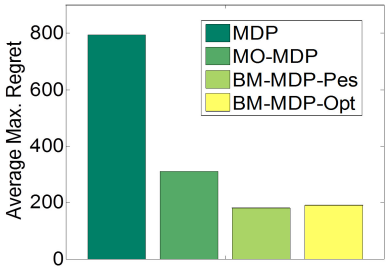


Figure 6: Performance of BM-MDPs

4.3 Real-world Test: Human Experiments

As a real-world test, we design and conduct a validation experiment on a pilot sample of participants (staff on campus). We conduct this investigation: i) to verify if SAVES can lead to changes in occupants' behaviors and to reduce energy consumption in commercial buildings, ii) to validate the parameter values used during the negotiation process such as the acceptance/compliance rate for the suggestion and iii) to understand what types of feedback are most effective to affect occupants' energy-related decisions.

In this study, we consider two test conditions: i) feedback without motivation (Test Group I) (e.g., please reduce the lighting level in your office), and ii) feedback with motivation including participant's own energy use, and environmental motives (Test Group II) (e.g., if you reduce your lighting for working hours, the annual energy savings at the building level are 26000kWh on average, which is equivalent to the reduction of CO_2 emissions of 2.2 homes for one year). From this experiment, we answer the following question by comparing change in energy behavior patterns and possible estimated energy consumption between test groups I and II.

HYPOTHESIS 1. *More informed feedback (provided to subjects in Test Group II) will be more effective to conserve energy than feedback without motivation (Test Group I).*

We tested the hypothesis above as follows: we first recruited 22 staff from 7 buildings at the University of Southern California who are over 18 years old. Subjects were tested under two different conditions, and each test group had 11 individuals respectively, each of whom has her/his own office. Since we tested using a simple lighting negotiation scenario, each participant must be able to adjust the lighting levels in her/his office. With participants' agreements, we installed lighting sensors (Figure 4) in their offices. During the experiment, participants were supposed to stay in their own offices and do their regular work. We then measured the baseline energy behavior and energy consumption, and SAVES provided feedback via emails based on sensed lighting level (two times per day, at 11am and 2pm, for three consecutive weekdays). In each message, participants received a simple suggestion for lighting level with a certain type of feedback (e.g., please reduce the lighting level in your office). We systematically observed and logged their energy behavior during the entire experiment using the light sensors. At the end of the experiment, each participant was required to take a short survey (i.e., the reasons why they agree or disagree with a provided suggestion). We conducted this study for two weeks in the fall of 2011 and collected data from human subjects using multiple sensors and routers.

Table 3: Lighting Negotiation Results (*: p < 0.05)

	Avg. Accep. Rate (%)	User Rating (Max: 5.0)
Group I	28.79 (11.03)	3.82 (0.26)
Group II	68.18 (9.65)	4.18 (0.18)
Mean Diff.	39.39*	0.36

In Table 3, column 2 displays the average acceptance rate in percentage (0–100%) of two test groups, and column 3 represents the average user rating of the provided feedback during the experiment. The range of ratings is between 0 and 5, and 0 indicates that the feedback was not helpful at all, and 5 means that the feedback was extremely helpful. In both columns, values in parentheses indicate the standard errors. The last row shows a mean difference between two groups for each value.

Table 3 shows that when we provided more informed feedback including environmental motives (Group II), occupants showed statistically significantly higher compliance acceptance rate (68.18%), which provides strong evidence for the above hypothesis (t-test; $p < 0.05$). In addition, human subjects in Group II felt that the provided feedback was more helpful during the negotiation process. However, the difference in user ratings between two groups was not significant, and thus we took a quick survey from participants at the end of the experiment to further analyze their decisions. In contrast with Group I, in Group II, the main reason why participants who agreed to reduce the lighting level in their offices (over 80% of conformers in Group II) was because the feedback significantly improved awareness of energy use. In addition, more than half of all participants strongly believed that this study will be very helpful by encouraging occupants to think about energy usage. This discrepancy in average user ratings and acceptance rates remains an issue for future work.

In this trial study, we have learned that although occupants in commercial buildings do not have a direct financial incentive in saving energy, proper motivations can achieve a higher compliance rate for the energy-related suggestion. This study specifically gives us the insights that there is a significant potential to conserve energy by investigating effective and tailored methods to improve occupants' motivation to conserve energy.

5. RELATED WORK

In discussing related work, a key point we wish to emphasize is the uniqueness of our work in combining research on multiagent systems, specifically our multi-objective MDP algorithm that handles uncertainty, and negotiations with human subjects, in an innovative application for energy savings. It is this specific combination of attributes that sets SAVES apart from previous research.

Multiagent Energy Systems: Multiagent systems have been considered to provide sustainable energy for smart grid management. Voice *et al.* [21] provided a game-theoretic framework for modeling storage devices in large-scale systems where each storage device is owned by a self-interested agent that aims to maximize its monetary profit. In addition, [10] addressed research challenges to integrate plug-in Electric Vehicles (EVs) into the smart grid.

To model and optimize building energy consumption, Ramchurn *et al.* [16] considered more complex deferrable loads and managing comfort in the residential buildings. Rogers *et al.* [17] addressed the challenge of adaptively controlling a home heating system in order to minimize cost and carbon emissions within a smart grid using Gaussian processes to predict the environmental parameters. Our domain is different in focusing on energy savings in commercial buildings, and the representation and approaches are also different from previous work by allowing consumers (i.e., occupants) to play a part in optimizing the operation in the building instead of managing the optimal demand on buildings.

Energy Literacy via Feedback: Abrahmase *et al.* [2] reviewed 38 interventions aimed to reduce household energy consumption, and they concluded that normative feedback about energy use is the most promising strategy for reducing and maintaining low consumption. However, it focused on residential environments, which is different from our work. In a recent study, Carrico and Riemer [4] provided monthly normative feedback via email to occupants of a commercial building about their own buildings' energy use in comparison with and other, similar buildings. Unfortunately, the study relied on self-reporting to assess the behaviors. Instead, our work relies on real sensors to observe their energy behavior in real-time. Faruqui *et al.* [7] reviewed past experiments and pilot projects to evaluate the effect of in-home displays (IHDs) on energy consumption. Our work is different because we simultaneously consider multiple criteria including energy consumption *and* occupant comfort level.

Multi-objective Optimization Techniques: There has been a significant amount of work done on multi-objective optimization. The most common approaches to multi-objective optimization are to find Pareto optimal solutions [15], use the weighted sum method to aggregate multiple objectives using a prior preference [22], or consider the weighted min-max (or *Tchebycheff*) formulation that provides a nice theoretical property in terms of sufficient/necessary conditions for Pareto optimality [14].

Recently, Chatterjee *et al.* [5] considered MDPs with multiple discounted reward objectives. They theoretically analyzed the complexity of the proposed approach and showed that the Pareto curve can be approximated in polynomial time. Ogryczak *et al.* [13] focused on finding a compromise solution in multi-objective MDPs for a well-balanced solution. They compared their approach relying on the Tchebycheff scalarizing function to the weighted sum method. On the other hand, there has been some significant advances to handle model uncertainty on standard MDPs including [6, 8]. Our work is different from them as we assume model uncertainty while simultaneously optimizing multiple criteria in MDPs.

6. CONCLUSION

In this work, we presented an innovative multiagent system called SAVES with the goal of conserving energy in commercial buildings. There are several key novelties in SAVES: (i) SAVES is based on a real building and uses actual building data, including energy data, occupant preferences and schedules; (ii) it investigates both individual and group negotiations to save energy in smaller offices and shared rooms; (iii) it focuses on a commercial building, which requires a different mechanism to effectively motivate occupants since they do not have a direct financial incentive in conserving energy; and (iv) SAVES's reasoning is based on a novel BM-MDP algorithm for generating optimal policies that explicitly considers multiple criteria optimization as well as uncertainty over occupant preferences. We justified SAVES in a validated simulation testbed and showed that it can provide solutions to significantly reduce energy consumption while achieving comparable satisfaction levels of building occupants. As a real-world test, we provided results of a human subject study where SAVES is shown to lead occupants to conserve energy in real buildings.

7. ACKNOWLEDGMENTS

We thank Perceptronics Solutions, Inc. for their support of this research.

8. REFERENCES

[1] *Buildings Energy Data Book*. U.S. Dept. of Energy, 2010.

[2] W. Abrahmase, L. Steg, C. Vlek, and T. Rothengatter. A review of intervention studies aimed at household energy conservation. *J Environ. Psychol.*, 25:273–291, 2005.

[3] S. Abras, S. Ploix, S. Pesty, and M. Jacomino. A multi-agent home automation system for power management. *Informatics in Control Automation and Robotics*, 15:59–68, 2008.

[4] A. Carrico and M. Riemer. Motivating energy conservation in the workplace: An evaluation of the use of group-level feedback and peer education. *J Environ. Psychol.*, 31, 2011.

[5] K. Chatterjee, R. Majumdar, and T. A. Henzinger. Markov decision processes with multiple objectives. In *STACS*, 2006.

[6] K. V. Delgado, S. Sanner, L. N. de Barros, and F. G. Cozman. Efficient solutions to factored MDPs with imprecise transition probabilities. In *AAAI*, 2009.

[7] A. Faruqui, S. Sergici, and A. Sharif. The impact of informational feedback on energy consumption - a survey of the experimental evidence. *Energy*, 35, 2010.

[8] R. Givan, S. Leach, and T. Dean. Bounded-parameter Markov decision processes. *Artificial Intelligence*, 2000.

[9] F. Jazizadeh, G. Kavulya, L. Klein, and B. Becerik-Gerber. Continuous sensing of occupant perception of indoor ambient factors. In *ASCE International Workshop on Computing in Civil Engineering*, 2011.

[10] S. Kamboj, W. Kempton, and K. S. Decker. Deploying power grid-integrated electric vehicles as a multi-agent system. In *AAMAS*, 2011.

[11] H. E. Khalifa, C. Isik, and J. F. I. Dannenhoffer. Energy efficiency of distributed environmental control systems. Technical Report DOE-ER63694-1, Syracuse Univ., 2006.

[12] Z. Mo and A. Mahdavi. An agent-based simulation-assisted approach to bi-lateral building systems control. In *IBPSA*, 2003.

[13] W. Ogryczak, P. Perny, and P. Weng. A compromise programming approach to multiobjective Markov decision processes. In *MCDM*, 2011.

[14] A. Osyczka. An approach to multicriterion optimization problems for engineering design. *Comput. Methods Appl. Mech. Eng.*, 15:309–333, 1978.

[15] V. Pareto. *Manuale di Economica Politica*. Societa Editrice Libraria, 1906.

[16] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. Agent-based control for decentralised demand side management in the smart grid. In *AAMAS*, 2011.

[17] A. Rogers, S. Maleki, S. Ghosh, and N. Jennings. Adaptive home heating control through Gaussian process prediction and mathematical programming. In *International Workshop on Agent Technology for Energy Systems (ATES)*, 2011.

[18] P. Scerri, D. V. Pynadath, and M. Tambe. Towards adjustable autonomy for the real world. *JAIR*, 17:171–228, 2002.

[19] N. Schurr, J. Marecki, and M. Tambe. Improving adjustable autonomy strategies for time-critical domains. In *AAMAS*, 2009.

[20] J. Tsai, N. Fridman, E. Bowring, M. Brown, S. Epstein, G. Kaminka, S. Marsella, A. Ogden, I. Rika, A. Sheel, M. Taylor, X. Wang, A. Zilka, and M. Tambe. ESCAPES - Evacuation Simulation with Children, Authorities, Parents, Emotions, and Social comparison. In *AAMAS*, 2011.

[21] T. Voice, P. Vytelingum, S. Ramchurn, A. Rogers, and N. Jennings. Decentralised control of micro-storage in the smart grid. In *AAAI*, 2011.

[22] K. Yoon and C.-L. Hwang. *Multiple Attribute Decision Making, An Introduction*. Sage Publications, 1995.

Active Malware Analysis using Stochastic Games

Simon A. Williamson
School of Information Systems
Singapore Management University
Singapore
swilliamson@smu.edu.sg

Pradeep Varakantham
School of Information Systems
Singapore Management University
Singapore
pradeepv@smu.edu.sg

Debin Gao
School of Information Systems
Singapore Management University
Singapore
dbgao@smu.edu.sg

Ong Chen Hui
DSO National Laboratories
Singapore
ochenhui@dso.org.sg

ABSTRACT

Cyber security is increasingly important for defending computer systems from loss of privacy or unauthorised use. One important aspect is threat analysis — how does an attacker infiltrate a system and what do they want once they are inside. This paper considers the problem of Active Malware Analysis, where we learn about the human or software intruder by actively interacting with it with the goal of learning about its behaviours and intentions, whilst at the same time that intruder may be trying to avoid detection or showing those behaviours and intentions. This game-theoretic active learning is then used to obtain a behavioural clustering of malware, an important contribution for both understanding malware at a high level and more crucially, for the deployment of effective anti-malware defences. This paper makes the following contributions: *(i)* A formal definition of the game-theoretic active malware analysis problem; *(ii)* A fast algorithm for learning about a malware in the active analysis problem which utilises the concept of reducing entropy in the beliefs about the malware; *(iii)* A virtual machine based agent architecture for the implementation of the active malware analysis problem and *(iv)* A behaviour based clustering of malware behaviour which is shown to be more accurate than a similar clustering using only passive information about the malware.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous

General Terms

Security, Experimentation

Keywords

Malware Analysis, Stochastic Game

1. INTRODUCTION

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

Cyber security has emerged as one of the most important problems in the modern internet age, with cyber attacks resulting in millions of pounds of damage to organisations and individuals. The rise of the internet has enabled the propagation of malicious software (malware), exposing home computer users and organisations alike to threats previously unimagined. Such threats include the stealing of users private data such as usernames, passwords and email contacts. These malware can also turn an unwitting computer users system into a tool for attacking the rest of the internet such as in Distributed Denial of Service attacks and the sending of spam. As a result, the problem then is of *Cyber Security* — how to defend a computer system against unauthorised use either by a piece of software or by a human attacker.

With this established, the aspect of cyber security considered in this paper is in threat analysis: Given a piece of malware or a human intruding on a system, can we learn about the behaviours and intentions of that intruder. This is also known as malware analysis. Here behaviours are taken to mean the software vulnerabilities exploited to gain access to some part of a system (such as opening a port or installing some keylogging software), and intentions are the goals of the attackers, do they want to steal some information or use the machine as part of a larger coordinated attack. This process has clear game-theoretic implications since intruders often want to mask their access in order that the attack can be used again. Consequently, we have the situation that we want to learn the maximum information possible about a malware on our computer system. Conversely that malware wants to avoid giving away information about it's behaviours and intentions. It is important to note that this is a central problem in cyber security, as the results of malware analysis are used to power virus detection.

Against this background, malware analysis is performed by a human security expert on each newly discovered malware. The security expert must use various tools to identify how the malware infiltrates a working system, how it propagates to other systems and what it does whilst on the infected system. This is a time consuming manual process during which the the expert will examine the malware binary, execute it, examine logs and make some trial interactions with the infected system. Due to the volume of threats that need to be analysed, several authors have proposed automated analysis techniques. Now, these techniques can generally be classified according to whether they are *static*, which means that the binary is analysed, or *dynamic*,

which means that the malware is executed and its effects monitored. Automated static analysis techniques include *Eureka* [6] which analyses the code and produces a control flow graph representing the malware logic. It does this by scanning for system calls and grouping them together and assigning functions (since several system calls are involved in a high level operation such as creating a file). However static analysis is becoming increasingly difficult as malware becomes more sophisticated at using techniques such as code obfuscation (where the binary is randomised but preserves the original logic) and polymorphic binaries (the code is mutated to change its identifiable features such as variable names). Consequently, dynamic analysis is an interesting avenue to consider further. Here the malware is actually executed and the results are analysed (which bypasses the problems with static analysis). Several automated analysis techniques have been proposed here including [1], [7] and [4] which gather execution traces (a list of the operations the malware performed) and then attempt to classify these traces. There are several means to classify the traces i.e. support vector machines or distance measures [2].

That said, we can identify a weakness with automated dynamic analysis techniques when compared with a human security expert. Specifically, all of these techniques are *passive*, meaning that the malware is executed and a log is generated. However, a human security expert would interact with an infected system by placing *honey* in several locations. Examples of this include creating fake Internet Explorer activity or sending emails. Malwares have unique responses to such activity which are missed by passive analysis. Consequently, we propose an automated technique based on *Active Dynamic Analysis*. This means that the system will interact with the malware, basing its next action on the response of the malware, with the aim of learning the maximum amount of information about the policy of that malware. There exist some steps in this direction such as [5] which acknowledges that some malware require an input and so they define a set of possible inputs to test. However, a weakness with this approach is that the input sets must be defined before execution and are not reactive to what the malware has done thus far. This makes this approach potentially slow since many unproductive paths may be explored. We will attempt to address this weakness by using software agents to *react* to what the malware has done and choose the next input. Table 1 summarises our classification of this space.

Table 1: Malware Analysis

	Dynamic	Static
Active	[5]	
Passive	[1], [7], [2], [4]	[6]

Finally, this paper makes the following contributions: *(i)* A formal definition of the game-theoretic active malware analysis problem; *(ii)* A fast algorithm for learning about a malware in the active analysis problem which utilises the concept of reducing entropy in the beliefs about the malware; *(iii)* A virtual machine based agent architecture for the implementation of the active malware analysis problem and *(iv)* A behaviour based clustering of malware behaviour which is shown to be more accurate than a similar clustering using only passive information about the malware.

2. MOTIVATING SCENARIOS

This section provides examples of active malware analysis on a single machine and on a network of machines. In the rest of this paper we will formalise these examples in terms of the *Active Malware Analysis Game*. We provide these illustrative examples to make concrete the process of active malware analysis and how it contrasts with simple passive analysis. Specifically, both of these examples will show that simply passively monitoring what a malware does is not guaranteed to find all aspects of that malwares behaviour and that many modern malwares only exhibit some actions in response to data or actions on the infected system, or even worse than that, may potentially try to evade analysis.

2.1 Malware acting on a Single Machine

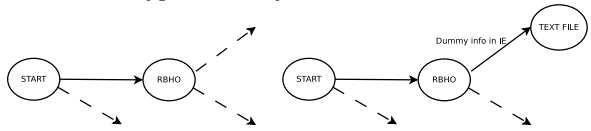
We first show the difference between passive and active analysis in the case of a single malware. Now, passive analysis:

*The malware **BZub.ji** is executed in a clean test environment. The subsequent trace is analysed by a technician and it is discovered that a browser helper object (RBHO) has been installed, which calls a new program placed in the Windows system directory. This program is analysed separately, but code obfuscation techniques render static analysis redundant. The program is executed but seems to be inert.*

By way of contrast, using active analysis techniques:

*The malware **BZub.ji** is executed in a clean test environment, alongside an analysis agent. The agent monitors that a browser helper object has been installed and that a secondary program has been installed. The agents model of existing malware indicates that these two activities indicate a modification of Internet Explorer has been used. The agent tests this aspect of it's model by executing a simulated interaction with Internet Explorer and it observes that a file has been created and is updated as it uses Internet Explorer. Simple textual matching shows that this file contains some of the honey that it used in IE. Consequently the agent has confirmed an aspect of the malware behaviour that passive analysis could only find with a human help.*

Figure 1 shows the difference in information received between the two types of analysis.



(a) Passive analysis (b) Active analysis

Figure 1: Analysis of the single malware scenario

2.2 Malware acting on a Network (Botnets)

The following example takes place in the context of a network of computer systems linked by some arbitrary topology. Assuming infiltration starts with a single machine, we first describe what we learn about the botnet using passive techniques and then contrast with active analysis:

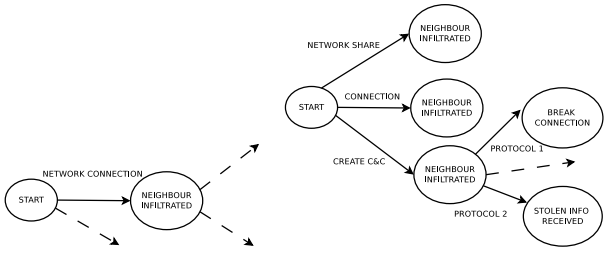
*The malware **Sinowal.aj** is executed on a single clean machine. Analysis of subsequent network traffic reveals that the infiltrated machine attempts to connect to a series of domains (presumably the command centre of the botnet). However no domains exist within the restricted network, so nothing further happens. Some connections to neighbours are recorded and the botnet grows larger by capturing these machines. The botnet takes no further action.*

This is contrasted with a more active analysis:

*The malware **Sinowal.aj** is executed on a single clean machine. The infiltrated machine attempts to connect to a series of domains, so a second machine on the network (with an analysis agent) poses as one of these domains and a connection is formed. This machine is now posing as the com-*

mand centre of the botnet. After some trial and error with the command protocol employed by the botnet, the machine is able to successfully communicate with the bot. Fake activity on the infiltrated machine causes it to connect to other machines on the network via shares and it is found that the malware can spread through these actions. Dummy information placed on these machines is also found to be harvested and sent to the command machine. Note that ***honey*** *can take the form of fake private information such as passwords or fake user activity such as opening a network share.*

Again Figure 2 shows the difference in information received between the two types of analysis.



(a) Passive analysis (b) Active analysis

Figure 2: Analysis of the botnet scenario

As we can see from these two examples the basic notion of active analysis remains the same, namely that more information about a malware can be found by interacting with an infected system (be it a single machine or network) than by just watching what a malware does. However the only difference between these two systems is the types of monitoring required (file system/ registry changes on a single machine verses network traffic in the botnet scenario) and the types of actions that can be taken (dummy Internet explorer activity versus connecting to network shares or sending emails). After establishing some basic notation we generalise these scenarios into the Active Malware Analysis Game.

3. BACKGROUND

This section describes the necessary background for the use of agent techniques in malware analysis. We first present a model of multi-agent interactions, Stochastic Games which can be used to model the game-theoretic interactions we presented in the previous section. Then we consider multi-agent learning within those games, the RMax algorithm which is capable of learning about the policies of unknown agents.

3.1 Stochastic Games

$\{N, S, A, \{R_j\}_{j \leq N}, T\}$ describes a discounted stochastic game:

- N is the set of players.
- S is the state space.
- $A = A_1 \times \cdots \times A_N$, represents the full set of actions, with A_j representing the set of actions for agent j.
- $R_j : S \times A \to \mathbb{R}$ is the reward function for agent j.
- $T : S \times A \times S' \to \mathbb{R}_{[0,1]}$ is the transition function.

The game is played as follows. At the first stage the game is in an initial state $s^1 \in S$. At stage m the players are informed of the past history $\left(s^1, a^1, s^2, \ldots, a^{m-1}, s^m\right)$, where s^t is the state of the game at timestep t and a^t is the action

Algorithm 1 The RMax algorithm.

```
1:  while true do                                          ▷ For each time step
2:      s ← CurrentState                                   ▷ Get current state
3:      a ← Action(s)                  ▷ Get best action for s using model
4:      r, s' ← Execute(a, s)                ▷ Reward and resulting state
5:      if Times played a in s ≤ Threshold then
6:          Update av(r, s, a)
7:          Update av(s', s, a)
8:          if Times played a in s = Threshold then
9:              R(s, a) = av(r, s, a)                      ▷ Set model
10:             T(s, a, s') = av(s', s, a)                 ▷ Set model
```

combination the players played at that state. Every player j chooses, independently of the others from its policy, π_i receives a stage payoff $R_j\left(s^m, a^m\right)$, where $a^m = \left(a_j^m\right)_{j \in N}$, and the game moves to a new state s_{m+1} according to the transition probability $T\left(s^m, a^m, s^{m+1}\right)$. Less formally, a stochastic game consists of a finite set of stage games between two or more agents. In each of these stage games, the agents can choose from a set of possibly unique actions, and depending on the choice made by all agents, are assigned a reward. Further to this, again depending on the actions chosen and the original stage game, a transition will occur to a new stage game, with possibly different actions and rewards.

The goal is to compute policies, $\pi_i : G \to A_i$ for all agents i at every time step t, such that no agent has an incentive to deviate. G indicates the history of observed states and actions of other agents. Put simply, this policy is a function which maps the history of the game to an action for agent i. The optimal policy returns the best action for that player.

3.2 The RMax Algorithm

RMAX [3] is an algorithm for learning an appropriate policy in a stochastic game. It assumes that the opponent is an initially unknown part of the environment, so it is suitable for single agent problems with an unknown underlying transition and reward function or multi-agent problems, such as ours, with an unknown opponent. The algorithm starts off with an optimistic model which assumes the maximum possible reward for all possible state actions. The learning procedure then proceeds by computing an optimal policy for this model and as states and actions become *known* (according to a polynomial threshold) updating the model and re-computing the policy. This algorithm is guaranteed to learn the policy for the agent in polynomial time. Figure 1 gives the algorithm in detail. The updates at lines 6 and 7 are the mean results of the previous trials. This purpose of using the average is to capture what happens when that action is taken both in the presence of a stochastic world model (which may result in different outcomes for the same action) or an opponent with an unknown policy (who may change which action she plays in the same state).

With this established, the next section builds on stochastic games in order to represent the unique characteristics of malware analysis on a computer system. Then in Section 4, we utilise RMax in the construction of a learning algorithm for active malware analysis which exploits those unique characteristics of the problem to learn quickly.

4. ACTIVE MALWARE ANALYSIS GAME

In this section we present the *Active Malware Analysis Game* between an analysis agent, n_1 and a malicious agent (malware) n_2. This formalisation captures, amongst others, the two scenarios presented in Section 2. The game specifies the interactions between a malware, who is trying to infiltrate a system, and an analyser who wants to learn about that

malware. In turn the malware may be trying to avoid such learning. The game is defined as follows:

- The infiltrated system is represented as a weighted graph where V is the set of vertices, E is the set of edges connecting the vertices. Each vertex is a state of core components of the system (ex: important flags in the registry) and edges represent transitions in the state of the computer that the malware can induce.
- The malware, n_2, may change several components of the infiltrated system. This represents a path through the graph past those vertices indicating the affected components. Thus the effects of the malware are represented by its location on the graph $v^m \in V$.
- The strategy space of the malware n_2 is the next change it can take from its current location, given as the neighbours of that location $a_2 = v^m \cup neighbour(v^m)$
- The analysis agent, n_1, may place within the system honey (simulated user activity). The set of locations of places honey can be placed is described by the subset of leaf nodes of the graph $V^H \subset V$.
- The strategy space of n_1, $a_1 = V^H$ means it can move the honey h from its current location, v^h, to a new location (or leave it where it is), $v^h \in V^H$. That is, the agent can create new simulated user activity and remove other activity. We use location on the graph to represent some fake user activity or data in place, with the preceding vertices representing state changes before this information is introduced.
- The global state space is given as the location of the honey, v^h, and of the malware, v^m, $S = V \times V^H$.
- Agent n_2 has a fixed, possibly stochastic, but unknown policy $\pi : S \rightarrow a_2$ which gives the probability of moving to a connected vertex given the location of the agent and the distribution of the honey on the graph.
- There is a reward function for agent n_1 associated with learning the policy and preferences of agent n_2. We will consider this further in the next section.
- We assume that the malware always starts out on a clean system at the start vertex $v^0 = s^1$.

If we refer back to the description of stochastic games, we can see that the graph represents a computer operating system and that vertices represent the different physical states that the operating system may be in (whether a certain registry variable is set or a type of file exists). Further to this, edges in the graph represent actions that the malware can take to change the condition of the operating system and exploit weaknesses (the transition function). Honey locations are the action space of the analysis agent. Consequently, it can be seen that a path from the start vertex to a honey represents a behaviour of the malware and we are interested in learning which behaviours a malware exhibits. We can see this in the Figures 1 and 2 where paths along the graph are an accumulation of the changes the malware has made, and that some edges are only taken in response to a particular honey (in these figures we represent this as the labels on the edges for simplicity).

Finally, it is clear that this is an instance of learning an opponents unknown policy in a stochastic game — both the analysis agent and the malware (be it software or a human agent) take sequential decisions and the reward function is linked because the analysis agent wants to learn the malware policy whilst that agent may want to hide its own policy.

5. LEARNING IN THE ACTIVE MALWARE ANALYSIS GAME

This section describes the learning algorithm we employ within the malware analysis game. Specifically, we describe several variants of the RMax algorithm that we later test in the empirical section. For these variants we compare their respective learning rates and finally describe how we represent learnt information for a user interested in generating a signature for the new malware.

5.1 Learning using Entropy Reduction

RMax assumes the optimal policy will be learnt with polynomial time. Now, in a real application such as ours, this is not feasible since placing a piece of dummy information on a computer system can take seconds. As a result, we are not interested in obtaining the eventual optimal policy, but in learning the most possible about the malware in very few timesteps (≤ 20). Consequently, we incorporate this notion into the reward function. However, we still want to employ RMax so that we can guarantee that the analysis agent is exploiting its knowledge of the malware effectively, whilst at the same time learning as much as possible. Given this, we define an information-centric utility function for the agent which can be used within RMax, and optimised in-order to learn about the malware as fast as possible.

Now, the malware policy π is defined as the distribution over the possible edges towards the honey (V^H) given the current location of the honey v^h and the malware position v^m. From the starting position v^0 to each of the honey locations v^H there is a path p_i which is defined as the list of edges $e_i \in E$ the malware will take to that honey $e_0, e_1, \ldots, e_n$. The malware then, must select a path $p_i \in P^H$ from the set of paths based on the location of the honey and its policy. The aim is to learn this policy which describes the probability of taking path p_i based on the location of the honey s, $Pr(p_i|s = v^H)$ for all possible honey locations.

$$Pr(p_i|s = v^H) = \prod_{e_n \in p_i} Pr(e_n|s = v^H) \quad (1)$$

where $Pr(e_n|s = v^H)$ is the probability of taking edge e_n.

We need a utility function which rewards the agent for learning this function online (since the initial value is an uninformative distribution). Consequently, we maximise the negative of the entropy of this function, where π_i is the current estimate of the malware policy:

$$U(\pi_i) = -[\sum_{s \in V^H} \sum_{p_i \in P^H} Pr(p_i|s) \log(Pr(p_i|s))] \quad (2)$$

With this established, the **MYOPIC** algorithm uses this utility function for 1-step lookahead action selection. At each timestep t the agent selects the action which maximises $U(\pi_t)$ and then updates π_t to a new policy π_{t+1} by updating the probabilities of taking an edge. Specifically the malwares *historical frequency* of playing edge e_n in s is defined as:

$$Pr(e_n|s = v_j^H) = \sigma_{e_n,s}^t = \frac{1}{t} \sum_{\tau=0}^{t-1} I\{e_n^\tau = s\},$$

where $I\{e'_n = e_n^\tau\}$ is an indicator function equal to one if e'_n is the action played by the malware at time τ, and zero

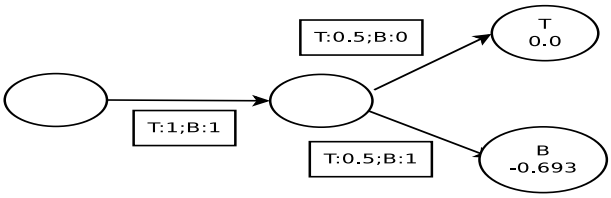


Figure 3: Myopic example.

otherwise. This algorithm works as follows: The example in Figure 3 shows a simple policy space for a dummy malware which starts in the first node attempts to move to the nearest honey, which can be at T or B. Previous trials have shown that the malware will steal information at B if it exists, but as yet we do not know what happens when information is at T, except that the malware moves to the second node in either case. Here we can see that if the honey is placed at B again, in this case (and assuming the malware tries to steal it), no new information is gained so the total entropy in the policy remains at -0.693. However, if the honey is placed at T then the entire policy can be learnt and entropy goes to zero, so *MYOPIC* would choose this action.

It should be noted that we abandon the optimality guarantees given by RMAX, however an optimal polynomial solution is not appropriate in our problem. Further to this, after presenting some benchmarks, the next section shows that this entropy reduction algorithm is guaranteed to be at least as fast as a random walk, and in general faster.

5.2 Benchmarks

PASSIVE: This algorithm does not take any action in response to the malware and represents the learning performed by dynamic, passive malware analysis such as [1]. This approach assumes that the malware will reveal its policy without interaction from the analysis agent. This allows us to benchmark our results against passive automated.
RANDOM: Selects a uniform distribution random action.
RMAX_OPTIMAL: Now, the aim of the problem is to learn the behaviour of the malware whilst she is changing the underlying system. As a result, this seems like a straightforward application of the RMax algorithm to the Active Malware Analysis Game. This is in contrast to using single agent learning algorithms which potentially ignore the problem that the malware may be trying to hide its policy from the analysis agent. However, this is potentially slow.

5.3 Exploration Rates

Now we define the exploration rates for our algorithms and justify the information-centric reward measure.

The single step expected entropy reduction $E[H]$ in the belief of the malware policy π_b is defined as:

$$E[H(\pi_b, v^m)] = \sum_{p \in P(v^m)} \sum_{v^h \in V^H} \pi(p,s) A(\pi_b, v^m, v^h) * [H(b(\pi_b, p, v^h)) - H(\pi_b)]$$

where p is a path from the set of all paths from the malware position v^m to the honey locations and $\pi(p, v^h)$ is the true malware policy and is the probability of taking path p in state v^h. $A(\pi_b, v^m, v^h)$ is the action selection function and is the probability of selecting honey location v^h for the current belief and malware position. Finally, $b(\pi, p, v^h)$ is the belief revision function giving a new belief π_b' when the malware takes path p in state v^h for prior belief π_b.

A random action selection policy is defined as follows:

$$A(\pi_b, v^m, v^h) = \frac{1}{|V^H|} \tag{3}$$

whilst the **MYOPIC** action selection gives:

$$A(.) = \begin{cases} 1 & v^h = h' \wedge argmax_{h'}[H(b(\pi_b, p, h')) - H(\pi_b)] \\ 0 & \text{otherwise} \end{cases}$$

Since **RANDOM** gives the expected entropy over all possible choices of $v^h \in V^H$ then this must include the v^h that would be chosen by **MYOPIC**. This means that we can decompose the expression for the expected entropy using **RANDOM** in terms of the expected entropy for **MYOPIC** and the expected entropy over all states not including that one chosen in **MYOPIC**. Now, we define J as the expected entropy using **MYOPIC**:

$$J = argmax_{v^h} \sum_{p \in P(v^m)} \sum_{v^h \in V^H} \pi(p, v^h)[H(b(\pi_b, p, v^h)) - H(\pi_b)] \tag{4}$$

where J_v is the v^h chosen in J which maximises the expression. Then remembering the choice of J_v, the expected entropy, using **RANDOM**, over the remaining set is:

$$\frac{|V^H| - 1}{|V^H|} \sum_{p \in P(v^m)} \sum_{v^h \in V^H_{-J_v}} \pi(p, v^h)[H(b(\pi_b, p, v^h)) - H(\pi_b)] \tag{5}$$

Now, let K represent the maximum entropy in the remaining set, $K = \sum_{p \in P(v^m)} \sum_{v^h \in V^H_{-J_s}} \pi(p, v^h)[H(b(\pi_b, p, v^h)) - H(\pi_b)]$ then the expected entropy for **RANDOM** is *at most* as large as:

$$\frac{1}{|V^H|} J + \frac{|V^H| - 1}{|V^H|} K \tag{6}$$

Putting all of this together,

$$\begin{aligned} RANDOM &\leq MYOPIC \\ \frac{1}{|V^H|} J + \frac{|V^H| - 1}{|V^H|} K &\leq J \\ K &\leq J \end{aligned}$$

Which means that as long as there is an v^h which is larger than all others, **MYOPIC** will reduce the entropy more quickly. If not, then it will do the same as **RANDOM**.

These expressions can be extended to a finite horizon n:

$$E[H_n(\pi_b, v^m)] = \sum_{p \in P(v^m)} \sum_{v^h \in V^H} \pi(p, v^h) A(\pi_b, v^m, s)[IH(\pi_b, p, v^h) + E[H_{n-1}(b(\pi_b, p, v^h), d(v^m, p))]]$$

where $IH(\pi_b, p, v^h) = H(b(\pi_b, p, v^h)) - H(\pi_b)$ and $d(v^m, p)$ is the malware transition from v^m along p to a new m'.

5.4 Learning over Multiple Malware

The techniques defined thus far are adequate for learning about a single malware, however they do not answer our larger research question: how similar is a new malware to an existing family of malwares? This is important in the context of using the information to power anti-virus defences. Specifically, by indicating a malwares similarity to existing malwares the process of generating a signature is simplified. This is the goal of all automated techniques. To address this, we will use the standard K-Means clustering approach which maintains a set of k families of malware together with

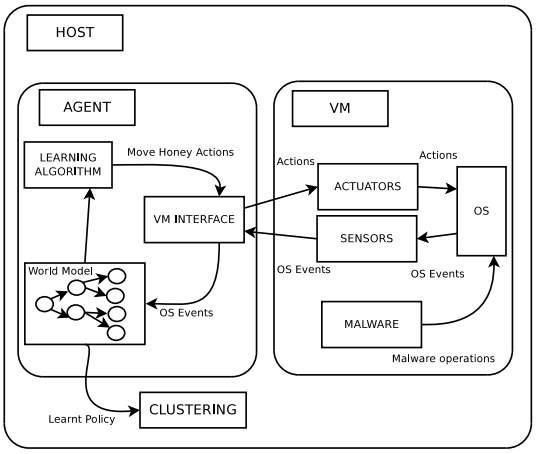


Figure 4: Active Malware Analysis Framework.

a representative mean malware policy. The goal then is to learn the policy of a new malware sample, and then assign it to an existing family or even create a new one, using a simple distance metric.

The distance metric takes the learnt policy and computes the distance for each transition (in all states). This is then summed to give a measure over the entire policy:

$$distance(\pi_i, \pi_j) = \sum_{v^h \in S} \sum_{p \in P(v_0)} |\pi_i(p, v^h) - \pi_j((p, v^h)| \quad (7)$$

6. ARCHITECTURE

This section describes the implementation of the Active Malware Analysis Game. The overall architecture is given in Figure 4. Since we are interacting with real examples of malware, we must run that malware binary (MALWARE) on a virtual machine (VM), and in order that the VM can be reset by the analysis agent (AGENT) must be located on the host system (HOST). Now, the agent makes use of several sensors to detect the state of the VM operating system (OS). Also, in order to interact with the malware, our agent needs access to a suite of actuators on the VM. Consequently we require an interface between the agent on the host and the sensors (SENSORS) and actuators (ACTUATORS) on the VM (VM INTERFACE). Next we give specific details of the actuators and sensors deployed in our empirical analysis.

6.1 Sensors

The Active Malware Analysis Game depicts the operating system state as a graph with vertices representing states for core components and edges are transitions between those states. Now, in order to detect the current state of the operating system and when such transitions occur (at the behest of the malware), we require a suite of sensors. Each sensor is designed to monitor one specific component such as whether a process has been registered to autorun when the operating system is started. For example this requires monitoring changes to the registry for the key: $\backslash Software\backslash Microsoft\backslash Windows\backslash CurrentVersion\backslash Run$. Similarly for other aspects such as browser helper objects, hidden services, parameters, and file system changes. By starting with the analysis performed by security experts on previous malware, we can generate a comprehensive set of such sensors, and the graph construction between them can be automated.

6.2 Actuators

The Active Malware Analysis Game allows the analysis agent to take actions in the operating system which the malware may or may not respond to. These actions allow the agent to move *honey* around the system. The purpose is to learn how the malware changes the operating system (using the sensors) in response to all of the possible honeys that might be deployed by the analysis agent. Consequently, our architecture requires a suite of honey actions. These include placing dummy sensitive information in several key locations or performing some simulated user activity on the operating system. One example of this includes opening Internet Explorer, going to a website and entering a username and password in fields denoted as such. Other examples include creating dummy configuration files for common programs or an address book of email address amongst others. In a similar fashion to the sensors, we start with types of honey identified by security experts to create a comprehensive suite.

As a final note, it can be seen that the architecture is easily expanded with new actuators and sensors should these be deemed necessary. The analysis agent will continue to learn as before with these new components and no change is required in the underlying algorithm.

7. EXPERIMENTS

In this section we demonstrate the utility of our active analysis framework by clustering a dataset of several previously analysed malwares. We show that the automatic clustering is accurate with regards to a human generated clustering and that it outperforms clustering performed using passive dynamic analysis. Further to this, we demonstrate that our entropy reduction learning algorithm is more useful in this malware analysis scenario than an RMax based algorithm. First we describe the experimental scenario, and then show the clustering performance over several algorithms.

7.1 Experimental Configuration

We experiment with a dataset of 50 malwares drawn from several families. These families are given in Table 2 All of these malware have previously been analysed manually and assigned to a cluster (both based on their static and dynamic properties). We allow each of our 4 learning algorithms ($RMAX\_OPTIMAL$, $MYOPIC$, $RANDOM$ and $PASSIVE$) to interact with each malware for 20 timesteps in a clean virtual machine. This is repeated 30 times and the average learnt policy is used in the clustering phase. Following this, we initialise the K-MEANS algorithm with 10 random means in the policy space and allow the clustering algorithm to run for 1000 iterations. This is repeated 30 times for the average clustering. Table 3 summarises the

Table 2: Experimental Malware

Zlob	Hooker	KeyLogger	LdPinch
PdPinch	QQPass	Sinowal	AdvanceKeyLogger
BZub	Luzia	VB	

types of behaviours and intentions we consider in this example (although the real set is larger). Now, the interesting thing to note about this set of malware is that some parts of their behaviour are conducive to passive analysis and some are more appropriate for active analysis. Specifically, many of the malware will install themselves in the system in various ways such as hidden services or injecting DLLs into other processes for example. Parts of this behaviour is identifiable by passive analysis because it always happens when

Table 3: Experimental Behaviours and Intentions

FSys/Rservice	Install service
Address	Emails
FSys/RBHO	Install BHO
Text	File contents
RAuto	Autorun
IE/Keylog	Private data from websites
RFile	Register file location
IE/Keylog/Cache	Private data from history

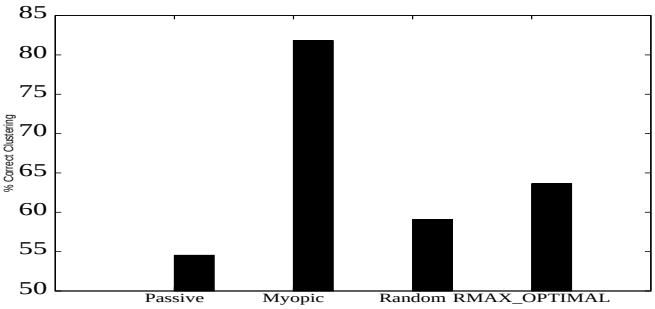


Figure 5: Cluster Identification Rate.

the malware executed. However some parts are not, such as when a keylogger writes a file in response to entering a username or inferring that an already running process has been captured by the malware.

7.2 Clustering

We first compare the clustering obtained by our various algorithms with an ideal clustering identified by malware analysis experts. As Figure 5 shows, the algorithm $MYOPIC$ is significantly more accurate when identifying clusters than $PASSIVE$, with a correct identification rate of 81% verses 54%. This is because $MYOPIC$ can identify a far larger part of the malware policy and consequently obtains a more informative clustering. Also, both $RMAX\_OPTIMAL$ and $RANDOM$ also outperform $PASSIVE$ because they all do active learning. However, the learning time is severely constrained and they do not learn as fast as $MYOPIC$ so consequently they are not as effective as that algorithm. We will show these results in more detail next to explain the improvement in performance.

Moving on, when we compare the features learnt in the malware policies we can see the impact on clustering to see why $MYOPIC$ performs much better than $PASSIVE$. Specifically, Figures 6, 8, 10 and 9 show the clustering for the algorithms $MYOPIC$, $PASSIVE$, $RMAX\_OPTIMAL$ and $RANDOM$ respectively. Further to this, Figure 7 shows the clustering done by a security expert. The x-axis shows the feature space for the possible mechanisms employed by the malware to infiltrate the system. The y-axis shows the possible locations of sensitive data that a malware might steal from. Both of these are in our restricted scenario. Each figure shows boxes for each identified cluster located in the space of mechanisms and intentions. The size of the box indicates the relative proportion of the corpus of sample malwares in this particular cluster.

With this established, in Figure 6 we can see that a large grouping of malwares installs a DLL as a system service and proceeds to keylog the users actions. The next smallest family registers an executable and steals data from text files.

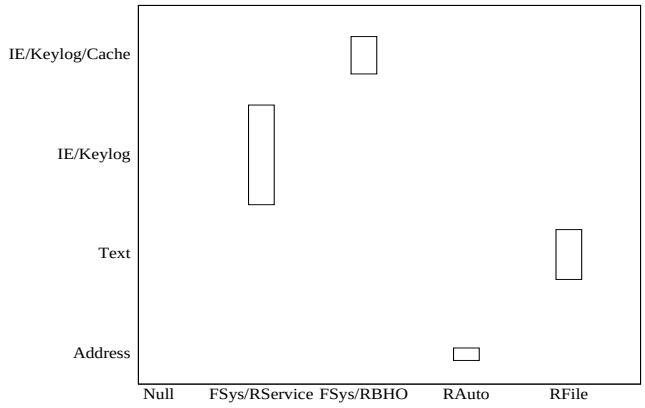


Figure 6: Feature extraction using $MYOPIC$.

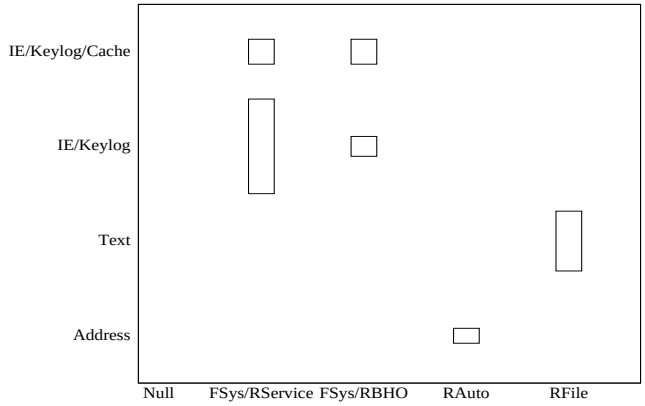


Figure 7: Expert classification.

An even smaller cluster registers a browser helper object and uses Internet Explorer to steal information. Finally a small group of malwares installs an autorun entry and raids the address book of Outlook.

As we can see in Figure 7, the clustering by $MYOPIC$ is very close to the one done by an expert. However the only divergence is in the large clusters which perform information stealing using keylogging and the cache at the same time. In some cases, $MYOPIC$ fails to differentiate between information stolen from the cache and from keylogging. This is because the malware in this case is able to perform both actions at the same time which breaks some of the underlying assumptions of a stochastic game. Further to this, sometimes there are some delays in placing the honey and when the malware reacts (perhaps because of errors in the malware or when it does polling) However, despite these physical limitations, as we will see next this clustering is still highly accurate compared to other approaches.

Specifically, the clustering from $MYOPIC$ should be contrasted with Figure 8 which has not learnt as much detailed information and so the clusterings are much courser. Here the algorithm typically can learn about how the malware is installed (a browser helper object or hidden service) but cannot find out about the intentions. An exception is the cluster of malwares that steals from text files - some of these files are created by the system and so are present even if a user does not create them. This is because, whilst the analysis

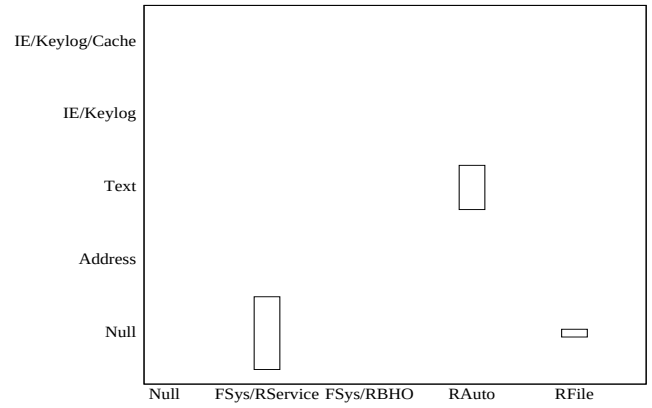


Figure 8: Feature extraction using $PASSIVE$.

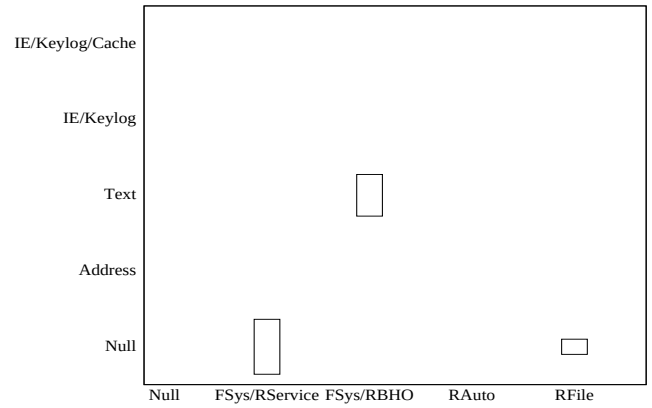


Figure 9: Feature extraction using $RANDOM$.

is dynamic (meaning the malware is executed), it is *passive*, meaning that we do not interact with the malware as a security expert tasked with analysis would. An illustrative example is the cluster of malwares which register a file: In Figure 6 we also see that this cluster does some keylogging, however in Figure 8 this information is missing and the clustering puts most of these malwares with other groups. As a result, automated analysis is limited in its usefulness unless it is *active* because many variants of malware require some user interaction to exhibit their full suite of behaviours.

Finally, we should consider what happens with active analysis using a slower learning algorithm - Figures 10 and 9. Here we can see that $RANDOM$ is effectively the same as $PASSIVE$ in the clustering it performs because it does not learn the important part of the policy space in the short time allotted. This highlights the importance of learning quickly in this domain. $RMAX\_OPTIMAL$ is better, and does in fact learn one of the clusters which requires active analysis (the cluster installing a keylogger in Internet Explorer), however even it does not learn the complete set of intentions for this cluster because it misses that this type of malware family also searches the cache.

8. CONCLUSION

This paper has introduced the application of Automated Active Malware Analysis using stochastic games and multi-agent learning. We defined a game capturing the active malware analysis problem. Following this, we developed an extension of RMax based on reducing entropy for learning quickly in the constrained time horizon of such games. We showed theoretically that this extension is faster than standard techniques. Finally, we presented a comprehensive empirical demonstration of our deployed framework for active malware analysis. We learnt the policies of 50 malwares and achieved a clustering very close to the one proposed by human security experts.

In future work, we intend to extend the application framework to the issue of learning about botnet malware. The game remains the same as in this paper, but a new architecture must be developed to monitor networks of systems, rather than the single system implemented in this paper. We also intend to extend the theoretical justification for the entropy reduction based algorithm, showing that it is faster than any other heuristic in this game.

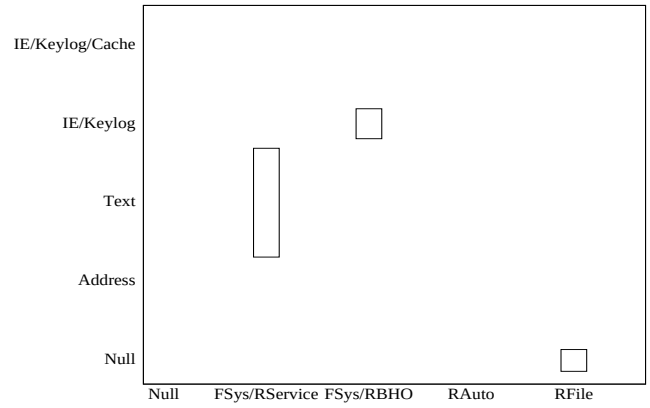


Figure 10: Using $RMAX\_OPTIMAL$.

9. ACKNOWLEDGMENTS

This work was supported by DSO National Laboratories, Singapore, contract DSOCL10192.

10. REFERENCES

[1] Michael Bailey, Jon Oberheide, Jon Andersen, Z. Mao, Farnam Jahanian, and Jose Nazario. Automated classification and analysis of internet malware. In Christopher Kruegel, Richard Lippmann, and Andrew Clark, editors, *Recent Advances in Intrusion Detection*, volume 4637 of *Lecture Notes in Computer Science*, pages 178–197. Springer Berlin / Heidelberg, 2007. 10.1007/978-3-540-74320-0_10.

[2] Ulrich Bayer, Paolo M. Comparetti, Clemens Hlauschek, Christopher Krügel, and Engin Kirda. Scalable, Behavior-Based Malware Clustering. 2009.

[3] Ronen I. Brafman and Moshe Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3:213–231, March 2003.

[4] Christopher Kruegel, Engin Kirda, Ulrich Bayer, and Andreas Moser. Dynamic analysis of malicious code. *Journal in Computer Virology*, 1 2006.

[5] A. Moser, C. Kruegel, and E. Kirda. Exploring multiple execution paths for malware analysis. In *Security and Privacy, 2007. SP '07. IEEE Symposium on*, pages 231 –245, may 2007.

[6] Monirul Sharif, Vinod Yegneswaran, Hassen Saidi, Phillip Porras, and Wenke Lee. Eureka: A framework for enabling static malware analysis. In Sushil Jajodia and Javier Lopez, editors, *Computer Security - ESORICS 2008*, volume 5283 of *Lecture Notes in Computer Science*, pages 481–500. Springer Berlin Heidelberg, 2008.

[7] GÃl'rard Wagener, Radu State, and Alexandre Dulaunoy. Malware behaviour analysis. *Journal in Computer Virology*, 4:279–287, 2008. 10.1007/s11416-007-0074-9.

Agents vs. Pirates: Multi-Agent Simulation and Optimization to Fight Maritime Piracy

Michal Jakob, Ondřej Vaněk, Ondřej Hrstka and Michal Pěchouček
Agent Technology Center, Faculty of Electrical Engineering, Czech Technical University in Prague
Technicka 2, Praha 6, Czech Republic
{jakob, vanek, hrstka, pechoucek}@agents.fel.cvut.cz

ABSTRACT

Contemporary maritime piracy presents a significant threat to the global shipping industry, with annual costs estimated at up to US$12bn. To address the threat, commanders and policymakers need new data-driven decision-support tools that will allow them to plan and execute counter-piracy activities most effectively. So far, however, the provision of such tools has been very limited. To fill this gap, we have employed the multi-agent approach and developed a novel suite of computational tools and techniques for operational management of counter-piracy operations. A comprehensive agent-based simulation enables the stakeholders to assess the efficiency of a range of piracy counter-measures, including recommended transit corridors, escorted convoys, group transit schemes, route randomization and navy patrol deployments. Decision-theoretic and game-theoretic optimization techniques further assist in discovering counter-measure configurations that yield the best trade-off between transportation security and cost. We demonstrate our approach on two case studies based on the problems and solutions currently explored by the maritime security community. Our work is the first integrated application of agent-based techniques to high-seas maritime security and opens a wide range of directions for follow-up research and development.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multi-agent Systems; I.6.3 [**Simulation and Modeling**]: Applications

General Terms

Algorithms, Security, Experimentation

Keywords

agent-based modeling, simulation, transportation, maritime piracy, security, optimization, case study, operation research

1. INTRODUCTION

Contemporary maritime piracy presents a significant threat to the global shipping industry, with annual total costs estimated at up to US$12bn [3]. International efforts have led to the reduction of the success rate of pirate attacks. The number of pirate attacks, average amount of ransom paid and the number of crewmembers held in captivity, however, remain high. In the first nine months of 2011 and for Somali-based piracy alone, there were 194 attacks reported for Somalia, 24 vessels hijacked and 400 crewmembers taken hostage[1].

From the many levels on which solutions to the problem are sought, we focus on the operational management of the situation at sea, as this is the arena where progress can be made in the short term, before long-term sustainable solutions can be developed onshore. Operational management is also the area where the agent-based approach can provide significant added value—due to the lack of strong central authorities, complex structure of shared, competing and hostile interests and fragmented, distributed nature, global maritime shipping is best viewed as a complex multi-agent system and, consequently, solutions pursued within the multi-agent framework.

In line with the efforts of main governmental and industry stakeholders, we focus on planning and management of operational piracy counter-measures designed to increase the security of maritime transit, including recommended transit corridors, group transit and escorted convoy schemes, route randomization and coordinated patrol deployments. When properly designed and implemented, these measures have a potential to significantly improve transportation security with reasonable additional cost. However, due to complex spatial and temporal dependencies between individual counter-measures, discovering their effective, synergistic combinations presents a major challenge.

We address this challenge in a novel way by a combination of agent-based modeling and optimization techniques. We have built a data-driven piracy-aware agent-based model of maritime transportation, which enables decision makers to reduce uncertainty about the effects of their management and regulatory actions. The model integrates a wide range of real-world data and, to our best knowledge, is the first computational model that simulates global shipping down to a level of individual vessels. This is crucial for accurately capturing emergent, collective effects that arise from the coordination and cooperation of commercial and navy vessels and their non-cooperative interaction with pirates.

Recognizing that being able to assess effects of piracy counter-measures is only a first step towards discovering their most effective combinations, we have also developed computational optimization techniques that partially auto-

[1]Source: IMB Piracy Reporting Centre

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

mate counter-measure design. We demonstrate the joint utility of the simulation and optimization tools on two case studies based on the problems and solutions currently explored by the maritime security community.

Please note that due to the wide scope of our research and the number of techniques developed, we only provide here a high-level description of key concepts, methods and components, with the prime focus on elucidating how the multi-agent perspective has been beneficial in addressing the problem at hand—references to papers studying individual issues in depth are provided where appropriate.

2. MARITIME TRANSPORTATION MULTI-AGENT SYSTEM

Maritime piracy takes place within a larger *maritime transportation system* comprised of all seaborne vessels engaged in maritime activities. At the level of abstraction suitable for operational management, the maritime transportation system can be viewed as a multi-agent system, with vessels corresponding to autonomous agents. Vessels in the system are capable of moving freely within the spatial boundaries of surface waters while interacting with the maritime environment, other vessels (either directly via communication or indirectly via environment) and other non-vessel actors (such as shipping operators or traffic coordinators). For most of the time, each vessel pursues its individual goals but there are also situations where multiple vessels interact. Such interactions are either non-cooperative (such as pirate attacks or navy warship pirate interceptions) or cooperative (such as merchant vessels' calls for help to navy warships).

2.1 Vessel Agents

The following types of vessels play the most important role in the dynamics of maritime piracy:

- *merchant ships* – large ocean-going vessels carrying cargo over long distances between world's ports; primary targets of pirate attacks;
- *pirate ships* – vessels of different types and sizes operating within and in close proximity to main shipping lanes, where they attempt to attack, board and hijack passing merchant vessels; depending on their operational area, pirate ships range from small skiffs up to large motherships acting as floating bases from which speedboats are launched to attack[2];
- *navy warships* – military vessels of different categories operating in piracy-affected areas and capable of armed action against pirates.

2.2 Piracy Counter-Measures

Merchant and navy vessels can participate in a range of piracy counter-measures designed to increase the security of voyage through piracy-affected waters. Most measures require cooperation between multiple vessels and can be viewed as multi-agent coordination mechanisms that complement standard, single-agent vessel behaviors. Based on discussions with the maritime security community, we consider the following operational piracy counter-measures:

[2]The group of one mothership and multiple accompanying speedboats is referred to as a *pirate attack group(PAG)*

Counter-measure	Parameters
Transit corridor	sequence of GPS waypoints
Patrol deployment	patrol stationary location and/or dynamic patrolling policy
Group transit	corridor, speed levels, transit schedule (per speed level)
Escorted convoy	corridor, departure time, convoy speed
Route randomization	corridor, randomization distribution

Table 1: Piracy counter-measures considered and sets of parameters by which they are specified.

- *Recommended transit corridors* concentrate merchant traffic along defined routes connecting sequences of waypoints. Corridors facilitate protection from navy vessels; however, they also makes targeting transiting vessels easier for pirates.
- *Group transit schemes* coordinate the timing of merchant vessel transit so that vessels pass high-risk piracy areas in groups; this improves mutual awareness and facilitates navy response; however, it makes the transit take longer as vessels have to follow a predefined entry schedule and may have to reduce their cruising speeds to match the speed of their respective transit group.
- *Patrol deployments* position navy warships in strategic locations from where they can provide assistance to nearby merchant vessels in case of a pirate attack; we consider both stationary and dynamic deployments. Navy patrols are very effective locally; however, their action radius is limited and their sustained operation carries huge costs.
- *Escorted convoy schemes* arrange incoming merchant vessels into escorted convoys prior to transiting a high-risk area. In contrast to the group transit scheme, the convoy is accompanied by a dedicated armed escort vessel throughout the transit. Although very effective, a large-scale convoy system would require a very high number of escort vessels to operate effectively, which is costly.
- *Route randomization* relaxes transit corridor boundaries by making transiting vessels randomly deviate from the center of the corridor according to a given probability distribution (typically uniform). Route randomization reduces predictability of vessel positions, and thus makes planning and execution of pirate attacks more difficult.

Each counter-measure can be parameterized by a set of parameters (see Table 1). Except for route randomization, all above measures are currently actively used, although convoy schemes are operated rather sporadically by national navies on an ad-hoc basis. The use of transit corridors and group transits is currently limited to the Gulf of Aden.

2.3 System Parametrization

From the perspective of maritime security, there are several factors fundamentally affecting temporal and geographical distribution of pirate attacks:

- *Merchant traffic patterns* – represented in terms of an *origin-destination matrix* describing a yearly number of trips between world's major shipping ports.
- *Pirate population* – represented by the number of active pirate groups and locations of their on-shore bases.
- *Weather conditions* – represented by spatio-temporal maps of wind direction and speed, sea state and visibility.
- *Piracy counter-measures* – represented by a list of specific counter-measures and their parameters.

The above factors comprise a minimum set of parameters that has to be specified for a maritime transportation system before its piracy-related properties can be studied, both in the real-world and in a simulation.

2.4 Performance Metrics

A wide range of events and measurable quantities can be observed on a maritime transportation system. On the system level, the following metrics are of practical interest with regards to evaluating and optimizing the efficiency and security of maritime shipping:

- *pirate attack statistics* – the number of pirate attacks that occurs in the system in a defined time period; we further distinguish between *successful attacks* (=hijacks), *intercepted attacks* (attacks that fail due to navy interception) and *aborted attacks* (attacks aborted by pirates themselves, often due to effective employment of self-defense measures by the targeted vessel);
- *average transit distance* – average distance travelled per merchant vessel trip (in kilometers);
- *average transit duration* – average duration of merchant vessel trip (in hours).

Additional metrics can be derived from the primary metrics, such as fuel consumption or operational cost per trip.

2.5 Problem Statement

Given the framework introduced above, the problem addressed by our work can be more precisely defined as:

1. Analyzing relationships between the parameters of the global maritime transportation system and its performance metrics
2. Discovering such combinations of piracy counter-measures that optimize a user-defined function over the performance metrics (i.e. a user-defined trade-off between the security and cost of shipping)

In the following two sections, we show how we addressed both problems.

3. SIMULATION

In order to address the first problem, we have built an agent-based model/simulation[3] of the global maritime transportation system. The simulation closely follows the multi-agent conceptualization of the transportation system described in the previous section. As such, it provides *executable* models of vessel behaviors as well as the (collective) piracy counter-measure.

Given the critical role the interactions between merchant, pirate and navy vessels play in the dynamics of maritime piracy, agent-based, micro-simulation approach is vital for accurately modeling the effect of piracy on maritime transportation, as it allows to capture phenomena and provide detail of analysis not attainable with macro-level equation-based methods [15].

3.1 Input Data

To achieve a sufficient level of accuracy, data-driven agent-based modeling requires large amounts of data for setting, calibrating and validating individual parts of the model. In contrast to macro-modeling approaches, data both on individual and macro level are required. To build the model of global maritime transportation, we have used the following categories of data: (1) *geographical data* (shore lines, shallow waters), (2) *weather data* (visibility range, sea state), (3) *merchant traffic data*[4] (origin-destination matrix, fleet composition, vessel operational characteristics), (4) *pirate intelligence* (base locations, attack strategies, capabilities, historic reported pirate incidents[5]), (5) *military operations* (number of warships and their operational areas) and (6) *piracy counter-measures* (see the previous section).

Obtaining the above data in quantity and quality required was and remains to be a major challenge due to enormous fragmentation of data gathering activities in the maritime domain and commercial sensitivity of some of key data. Moreover, many data sets obtained were noisy and incomplete and required significant preprocessing before integration to the model.

3.2 Agent Behavior Implementation

Maritime transportation simulation requires agent control architecture capable of expressing desired individual and collective vessel behaviors. Agents have to be able to execute long-running actions while reacting to interruptions. The minimum intelligent agent architecture that can handle such requirements is a model-based reflex agentwith an encapsulated deliberative module handling route-planning. The required class of behaviors should be implementable in a modular and extensible way, facilitating sharing of common behavior fragments between different classes of vessels. At the same time, the agent control architecture should be computationally efficient enough to handle thousands of simulated agents. Unfortunately, none of existing agents architectures or simulation platforms supports these requirements. General agent-based simulation toolkits do not provide sufficient abstractions for modular behavior implementation; such abstractions are provided by cognitive agent architectures (e.g. Jazzyk [10]) but these require substantial computational resources to run.

Our implementation therefore uses *extended finite state machines (FSM)*, which augment standard FSMs with internal variables associated with each state. Individual states correspond to the principal mental states of the vessel agent

[3]We use both terms rather interchangeably, choosing *model* when focusing on the descriptive aspect and choosing *simulation* to emphasize the dynamic/execution aspect

[4]Merchant vessel trajectories in the form of AIS records are available at e.g. AISLive: `http://www.aislive.com/`

[5]Pirate reports are available from e.g. OceanusLive: `http://www.oceanuslive.org`

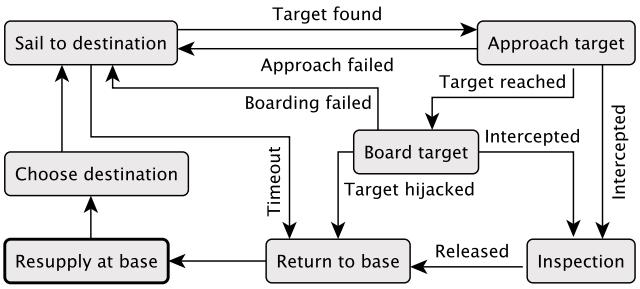


Figure 1: A finite state machine of the pirate vessel agent.

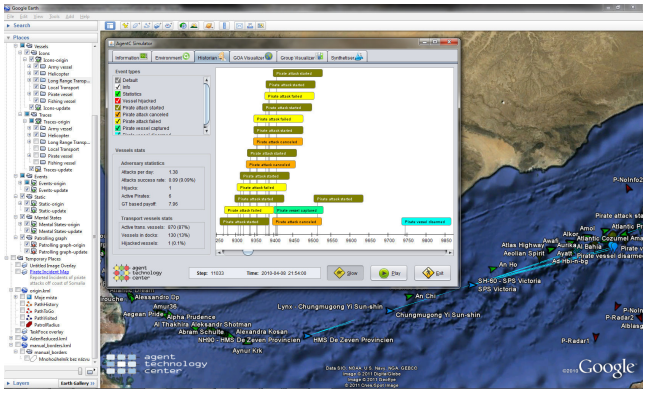


Figure 2: AGENTC simulation platform—visualization of the Gulf of Aden group transit.

(such as move, attack, patrol etc.) and their associated actions. These actions may involve complex deliberative procedures, e.g. risk-aware route planning for merchant ships and adaptive target selection for pirates, giving the FSMs reasoning capabilities beyond simple reactive control. Although limited (e.g. not capable of executing concurrent activities), extended FSMs proved to provide a good trade-off between expressivity/modularity and computational efficiency. An example FSM is given in Figure 1.

3.3 Simulation Platform Implementation

Extended FSM-based behavioral models are executed on a Java-based multi-agent platform built partially using the lightweight ALITE [6] multi-agent simulation toolkit and employing Google Earth for geo-spatial visualization (see Figure 2). The simulation platform provides abstractions for representing the maritime environment, agent-to-environment sensor interfaces and agent-to-agent communication protocols. Time-stepped simulation execution model is used, although we consider transitioning to the event-based model to further improve computational efficiency. Each simulation run is defined by a scenario defining the parameters of the maritime transportation system (Section 2.3). Parallel execution of large numbers of simulations is supported using the Eucalyptus cloud-computing platform[7].

3.4 Model Validation

The model has been validated both on the individual and system level. Individual-level models have been validated against independent test datasets capturing real-world behavioral patterns for the respective type of vessels. For example, merchant vessel models have been validated against real-world trajectories obtained from satellite AIS and voluntary reporting systems. In addition, system-level behavior has been validated against empirical data. The spatial distribution of merchant traffic has been compared with maritime shipping density maps. To validate the interaction of all types of vessels and counter-measures employed, we compared pirate incidents generated by the model with real-world piracy incidents. More details about the validation can be found in [17].

[6] `http://agents.fel.cvut.cz/projects#alite`
[7] `http://www.eucalyptus.com/`

4. OPTIMIZATION

The maritime transportation simulation introduced in the previous section empowers policy-makers in estimating the effects of different combinations of piracy counter-measures. Given the number parameters and combinations of these counter-measures, however, finding their right configuration remains difficult. We have therefore explored ways to provide computational support for optimizing counter-measure configurations automatically.

In its full generality, looking for such optimum configurations is a massive multi-agent optimization problem: a multi-objective performance function is optimized in a stochastic and partially observable environment with a high degree of uncertainty and thousands of interacting, largely self-interested agents employing a wide range of parameterized strategies and policies in the presence of multiple adaptive adversaries. Even if solutions in a form of a massive joint transit and patrolling routes and schedules could be found, they might be too complex and unstructured to be understood and hence trusted by human stakeholders. Instead of trying to solve the full problem, we therefore focused on optimizing individual counter-measures. For computational reasons, proposed optimization algorithms work with highly abstracted problem representations not containing the same amount of details as the simulation model. The simulation is therefore used to validate optimization results and to obtain higher-accuracy assessment of their expected real-world performance, which can be subsequently used to fine-tune proposed solutions.

4.1 Group Transit Optimization

Since participation in group transit schemes is voluntary, the problem of determining optimum speed levels and transit schedules can be viewed as a cooperative game with non-transferable utilities. Because of computational intractability of solving such a game for real-world problem sizes, we have so far considered two simplified, cooperative formulations of the problem.

The simpler formulation concerns the optimization of *fixed-schedule* group transit schemes. Taking into account the distribution of cruising speeds of transit traffic, we look for such a fixed set of speed levels that result in a shortest average transit duration. The optimum set of speed levels is found by searching for an optimum binning of the histogram of transit cruising speeds using a branch&bound search combined with dynamic programming (see [6] for details).

The more advanced formulation explores the potential of *dynamic* group transit schemes, in which speeds and schedules are not fixed in advance but determined on-the-fly, using multi-agent coordination techniques, based on incoming

traffic. The added flexibility allows the dynamic group transit scheme to achieve higher performance than fixed-schedule schemes on the expense of more extensive vessel coordination and information sharing.

4.2 Randomized Transit Routing

A major disadvantages of fixed transit corridors is the high predictability of vessel positions [13], which makes targeting merchant traffic easier for pirates. Predictability can be reduced by instilling a certain amount of randomness in transit routing. A basic approach applies uniformly or normally distributed randomization to disperse the traffic away from the corridor center and/or to alternate between several predefined corridors. Better route randomizations can be obtained using game-theoretic techniques, which explicitly account for the payoff the merchant vessels and attacking pirates receive from different transit routes. To this end, we extended the model of security games and formalized the hostile area transit problem as a zero-sum normal-form game between two mobile players, the transit and the pirate, each choosing a route maximizing its utility. The solution, found using incremental equilibrium search techniques, can achieve up to two-fold reduction in the attack rate compared to the basic approach. Details are provided in [16, 18].

4.3 Optimum Patrol Deployment

Due to their limited numbers and very high operational costs, military vessels have to be deployed in a way maximizing their protective effect. We considered two formulations of patrol deployment problem. The basic formulation considers a *stationary* deployment of patrol vessels—each vessel is assigned a fixed location from which it only departs to assist vessels in danger. Given a number of patrol vessels, we look for such a set of deployment locations that maximizes the volume of commercial traffic within the patrol's effective action radius[8]). Deployment locations are found using a vector quantization *Lindo-Buzo-Gray (LBG) algorithm* [9]. See Figure 3 for an illustrative example.

The basic formulation does not take into account the ability of pirates to adapt their attack locations in response to their observation of fixed patrol deployments. To overcome this problem, we therefore considered a game-theoretic formulation of the problem in which patrols are mobile and randomize their movement to minimize the ability of pirates to take advantage of their predictable absence, while taking into account transit routes of individual merchant vessels. An novel extended-form game in the Stackelberg setting is used as the underlying formal model. The approach is particularly effective in combination with randomized transit routing. See [2] for more details.

5. CASE STUDIES

We have applied the developed modeling and optimization tools to several real-world use cases, based largely on feedback from the maritime community and discussions with officials from the International Maritime Organization and U.S. Office of Naval Research. Here we present two specific case studies—one focusing on evaluating the combination of a novel corridor system and patrol deployments in the piracy-infested Indian Ocean, the other on the optimization of group transit scheme in the Gulf of Aden. More information about the developed tools and their applications can be found at the AGENTC project website[9].

[8]Radius of 150km is currently considered; this corresponds to the ability to respond within 40 minutes using an onboard helicopter.

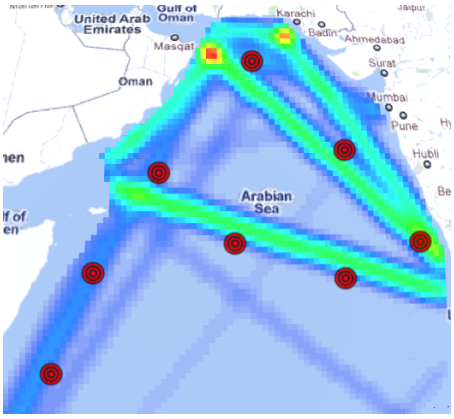


Figure 3: Example traffic density and the corresponding traffic coverage-maximizing deployment of 8 patrols.

Except for the parameters explicitly mentioned as variables of the study, the maritime system configuration remains the same throughout both case studies, in particular the origin-destination matrix capturing global merchant shipping flows. The simulation contained approximately 4500 merchant vessel agents, up to 100 navy warships and up to 20 pirate ship agents. In both case studies, the results given are for one year of simulated maritime traffic. Because parts of agent decision making are inherently nondeterministic, each configuration was simulated for 100 runs and average values are presented. One simulation run took approximately 10 mins of single 2.5GHz CPU core execution time.

5.1 CS I: Indian Ocean Corridor System

The *International Recommended Transit Corridor (IRTC)*, established in 2009, has since proven—in combination with the deployment of navy patrols—a very effective tool for suppressing successful pirates attacks in the Gulf of Aden. Recently, the maritime security community has been discussing the possibility of establishing additional corridors in the Indian Ocean, where pirate activity is also high following pirates' displacement from the Gulf of Aden. In contrast to the Gulf of Aden, which is an elongated, narrow area with a simple bidirectional traffic flow, the Indian Ocean is much larger and crisscrossed, in all directions, by a multitude of traffic flows. This makes the design of an effective corridor system a complex optimization task.

Scenarios. We used the AGENTC simulation to study three possible layouts of Indian Ocean corridor systems: (1) single *west-east corridor* channeling the large amount of west- and east-bound traffic (denoted as *Single-IO*), and (2) a more extensive *multi-corridor system* covering all the main traffic flows in the Indian Ocean (denoted as *Multi-IO*). Results are compared with the current setup where no corridors are used in the Indian Ocean (denoted as *None-IO*). IRTC corridor is considered in all cases. See Figure 4 for corridor layouts.

[9]`http://agents.fel.cvut.cz/projects/agentc`

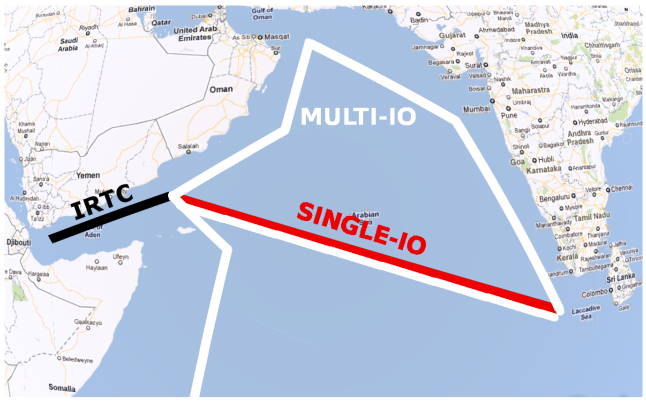


Figure 4: Corridor layouts for the Indian Ocean corridor system. The Single-IO layout uses the IRTC and the red east-west corridor only; the Multi-IO layout utilizes all depicted corridors.

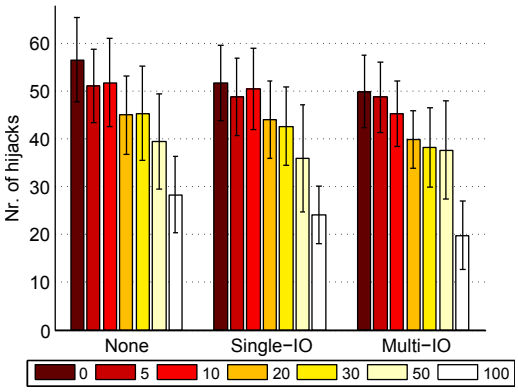


Figure 5: Dependency of the number of hijacks on the corridor system and the number of patrols (0–100). Standard deviation over 100 simulation runs also depicted.

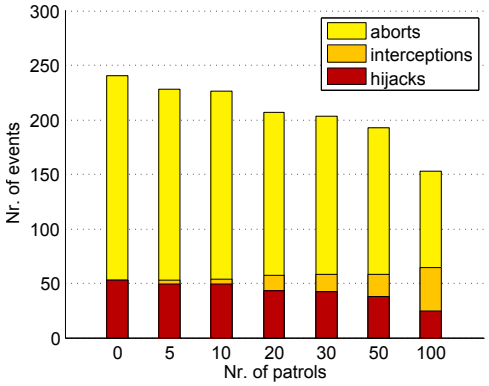


Figure 6: Ratios of *hijack*, *abort* and *interception* outcomes of pirate attacks for different numbers of patrol vessels.

In addition, we were interested in better understanding the possible synergy of deploying navy vessels alongside transit corridors. As a second study parameter, we therefore varied the number of deployed navy warships, using the stationary deployment method to determine their positions (see Section 4.3). All results presented are for three active pirate attack groups.

Results. We have evaluated all performance metrics defined in Section 2.4. Average transit distance and duration only depend on the corridor system and amounted to 6696km / 237h for no corridors in the Indian Ocean, 6703km / 237h for the Single-IO and 6819km / 242h for the Multi-IO corridor setup.

Pirate attacks statistics depend on both study variables. The numbers of hijacks in Figure 5 confirm the synergistic effect of transit corridors and patrolling—the Multi-IO corridor setup boosts protection force of patrols up to 40% in the case of 100 patrols (28.9 vs. 20.4 hijacks for None-IO and Multi-IO corridor system, respectively). A detailed breakdown of attack outcomes for the multi-IO corridor configuration (Figure 6) indicates that the decrease in hijacks is both due to the warship deterrence effect and the ability to intercept pirate attacks if they actually take place (more of the latter as the number of patrols increases). Finally, in Figure 7 we compare geographical distribution of vessel hijacks for None-IO and Multi-IO corridor setup with 20 patrolling warships. The distribution clearly depicts a high-risk hotspot north-east of the Socotra island.

Overall the results suggest that establishing a transit corridor system in the Indian Ocean is an effective way of increasing the security of transit on the expense of a very small increase in transit distance and duration (about 2% in the case of Multi-IO configuration). Further improvements might be attained if corridors are combined with group transit and/or escorted convoy measures, though that could have a noticeable impact on transit duration.

5.2 CS II: GOA Group Transit Optimization

In August 2010, the Group Transit Scheme was introduced to further reduce the risk of pirate attacks on vessels transiting the Gulf of Aden. The scheme, which is closely related to the IRTC, groups vessels traveling at similar speeds so that they cross high-risk areas close together as this provides additional deterrence to pirates and facilitates military response in case of an attack. Each transit group follows a recommended route through the IRTC at a published speed and fixed schedule (see Figure 8a) designed to maximize protection in highest risk times and areas. The schedule specifies vessel entry times depending on vessel's cruising speed. Five speed levels and consequently five speed groups are currently used—10, 12, 14, 16, and 18 knots.

The current number of speed levels and their uniform spacing is not optimum, given the distribution of cruising speeds in typical transit traffic (see Figure 8b). The aim of this study thus was to find out whether a different distribution of speed levels could reduce the transit delay incurred by following the group transit schedule.

Scenarios. The study variables were the number and distribution of speed levels used by the Gulf of Aden group transit scheme. For each number of speed levels, their optimum distribution was determined as the one maximizing the average speed of transit (and thus minimizing average transit time).

In addition to proposing optimum speed level distribution for fixed-schedule group transit, we have also explored the potential of a dynamic, negotiation-based group transit scheme in which groups are formed on-the-fly as vessels arrive.

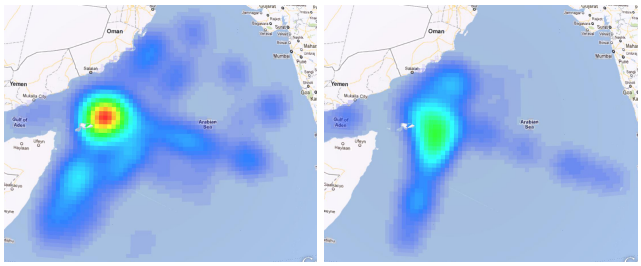


Figure 7: Geospatial distribution of hijacks for None-IO (left) and Multi-IO (right) corridor system configurations.

Results. We used the simulation to evaluate all key performance metrics. Surprisingly, no significant shift in the number of hijacks was observed. Upon a closer analysis, this can be explained by two forces cancelling each other—on one hand, a higher number of speed levels results in higher average transit speeds which reduces the attack success rate; on the other hand, more speed levels means more transit groups which makes their protection more difficult.

Expectedly, the average transit delay decreases with increasing the number of speed levels (Figure 8c). For the approximately 22 thousands vessels transiting the Gulf of Aden every year, the optimized 6 speed-level transit scheme would save over 400 days of total transit time a year. Considering an average daily vessel operational cost of US$30K, this translates into savings of approximately US$12mn a year. The dynamic group transit scheme surpasses even the best fixed-schedule scheme both in the number of hijacks and the transit time (the red dashed line in Figure 8c). Practical application of dynamic group transit would, however, require more extensive changes to the way transit is organized—a rather symptomatic trade-off between the performance and practicality of piracy counter-measures (see also the next section). See [17] for a more detailed evaluation and discussion of the case study.

6. LESSONS LEARNED

Overall, the multi-agent paradigm proved very useful during all stages of the development process—providing a conceptual framework for the analysis of the problem, informing architectural decisions during system design and, finally, supplying modeling and optimization techniques to implement the required functionality. Many challenges were not technical; often they lied in the (in)ability to obtain essential domain knowledge and datasets.

That said, the development of the simulation would have been easier if there was an agent-based simulation platform offering higher-level abstractions for representing individual- and collective level behavior and capable of simulating thousands of agents simultaneously. As mentioned in Section 3.2, to our best knowledge, no such a domain-independent platform currently exists, despite the fact that it would be useful in a wide range of applications. The optimization part would benefit from further research on (route) planning, scheduling and security resource allocation in hostile settings. Existing techniques, grounded largely in computational game theory, remain limited in their scalability and reliance on strong assumptions concerning rationality of adversaries and observability of their actions, although there is promising recent work on their relaxation (e.g., [7]).

As far as pitching of the agent-based approach is concerned, the ability to visualize the execution of individual simulation runs proved vital. First, it aided in conveying the very idea of maritime transportation as a multi-agent system. Second, it helped in winning the confidence of domain experts by allowing them to peek inside the model (see Figure 2) and verify that it behaves realistically on the micro-level. A key selling point of the agent-based approach to analyzing counter-piracy measures was the ability of the approach to analyze *hypothetical what-if* scenarios not yet occurring in the real world. Such scenarios cannot be reliably explored using standard statistical and/or data mining methods because they are too different from existing real-world situations on which datasets required for generating such models can only be obtained.

Working with the user community, we were constantly reminded of the necessity to maintain a proper balance between the quest for sophisticated, optimum solutions and their suitability for real-life deployment. Simple, suboptimal yet robust solutions can often be more suitable not only because they rely on fewer uncertain assumptions (such as rationality, observability or information sharing), but also because they are easier to explain and compatible with the existing infrastructure and processes in the generally very conservative maritime domain. Rather than coming up with a revolutionary optimum ways of managing counter-piracy measures, a more evolutionary approach seems more suitable, starting from concepts and measures already familiar to domain experts and using sophisticated techniques to discover their optimum configurations.

In contrast to other transportation domains, (global) maritime shipping seems underrepresented in the applied research on multi-agent systems and, in fact, on computational modeling and optimization in general. This is despite the fact that the global, transnational nature of maritime shipping and the consequent lack of strong central authorities, complex incentive structure and stiff economic competition make the multi-agent framework indispensable for accurately representing global shipping problems. Given the size of the shipping industry, estimated at several hundred US$ billion annually, current situation presents a sizable opportunity for innovative applications of multi-agent techniques.

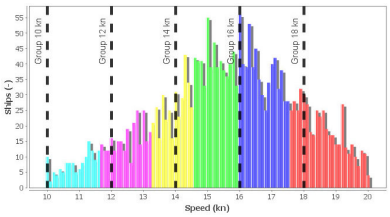
7. RELATED WORK

Unlike in other areas of transportation, most notably road and air transportation, the deployment of computational modeling in the maritime domain is limited. Existing work either focuses on traffic in ports and national, coastal waters [8, 5] or uses high-level equation-based models [1] unfit for capturing individual-level behavior and inter-vessel interactions essential to model maritime piracy. The relative lack of work addressing global shipping as a whole is partly due to the global, international nature of maritime shipping and the consequent lack of a strong authority that would drive implementation of such methods.

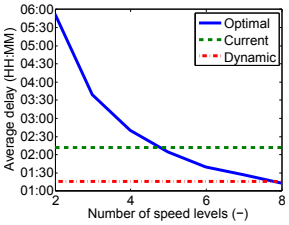
Focusing on the very phenomenon of maritime piracy, the work is even more slim and concentrated primarily in the fields of security studies, international relations and global policy (e.g. [11]). Only very recently, initial attempts at applying computational modeling and optimization to maritime piracy have emerged but focus exclusively at military aspects of the problem [14, 12, 4].

Speed	Entry point A – time	Entry point B – time
10 kts	04:00 GMT+3	18:00 GMT+3
12 kts	08:30 GMT+3	00:01 GMT+3
14 kts	11:30 GMT+3	04:00 GMT+3
16 kts	14:00 GMT+3	08:30 GMT+3
18 kts	16:00 GMT+3	10:00 GMT+3

(a) Current group transit schedule.



(b) Transit vessel speed distribution.



(c) Average transit delay.

Figure 8: (a) Schedule used by the current Gulf of Aden group transit scheme, (b) histogram of transit traffic speeds and its current (dashed lines) and optimum (colors) binning (for 6 speed levels), (c) reduction of the average transit delay with the increasing number of speed levels; delay for the current suboptimum (green) and dynamic grouping (red) schedule also shown.

8. CONCLUSIONS

We have shown how the multi-agent approach can be used to conceptualize and consequently address important challenges in planning and managing counter-piracy activities. A combination of multi-agent simulation and optimization proved very useful, enabling to evaluate and optimize the performance of counter-measures in a wide range of what-if scenarios. To our best knowledge, our work is the first integrated application of agent-based techniques to high-seas maritime security and, in fact, to global shipping analysis and optimization in general. The techniques developed enable commanders, policymakers and other relevant stakeholders to make better, more informed decisions and to improve maritime security with reasonable additional cost.

Acknowledgements

Funded by the Office of Naval Research (grant no. N000140 910537) and by the Czech Ministry of Education, Youth and Sports (grant no. LH11051).

9. REFERENCES

[1] S. Bourdon, Y. Gauthier, and J. Greiss. MATRICS: A maritime traffic simulation. Technical report, Defence R&D Canada, 2007.

[2] B. Bošanský, V. Lisý, M. Jakob, and M. Pěchouček. Computing time-dependent policies for patrolling games with mobile targets. In *10th International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[3] A. Bowden, K. Hurlburt, E. Aloyo, C. Marts, and A. Lee. The economic costs of maritime piracy. Technical report, One Earth Future Foundation, 2010.

[4] J. Decraene, M. Anderson, and M. Low. Maritime counter-piracy study using agent-based simulations. In *2010 Spring Simulation Multiconference*, page 165, 2010.

[5] K. Hasegawa, K. Hata, M. Shioji, K. Niwa, S. Mori, and H. Fukuda. Maritime traffic simulation in congested waterways and its applications. In *4th Conference for New Ship and Marine Technology, Chine*, pages 195–199, 2004.

[6] O. Hrstka and O. Vaněk. Optimizing group transit in the Gulf of Aden. In *15th International Student Conference on Electrical Engineering*, 2011.

[7] D. Korzhyk, V. Conitzer, and R. Parr. Solving Stackelberg games with uncertain observability. In *10th International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[8] E. Köse, E. Basar, E. Demirci, A. Güneroglu, and S. Erkebay. Simulation of marine traffic in Istanbul strait. *Simulation Modelling Practice and Theory*, 11(7-8):597–608, 2003.

[9] Y. Linde, A. Buzo, and R. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, 28(1):84–95, 1980.

[10] P. Novák. Jazzyk: A programming language for hybrid agents with heterogeneous knowledge representations. *6th International Workshop on Programming Multi-Agent Systems*, pages 72–87, 2009.

[11] F. C. Onuoha. Piracy and maritime security off the Horn of Africa: Connections, causes, and concerns. *African Security*, 3(4):191–215, 2010.

[12] L. Slootmaker. Countering piracy with the next-generation piracy performance surface model (master thesis). Technical report, Naval Postgraduate School, Monterey California, 2011.

[13] F. J. Sluiman and H. de Konig. Naval vessel traffic services: Enhancing the safety of merchant shipping in maritime security operations. *Naval War College Review*, 63(3):123–137, 2011.

[14] T. Tsilis. Counter-piracy escort operations in the Gulf of Aden (master thesis). Technical report, Naval Postgraduate School, Monterey California, 2011.

[15] H. Van Dyke Parunak, R. Savit, and R. Riolo. Agent-based modeling vs. equation-based modeling: A case study and users' guide. In *1st International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 277–283, 1998.

[16] O. Vaněk, B. Bošanský, M. Jakob, and M. Pěchouček. Transiting areas patrolled by a mobile adversary. In *2010 IEEE Symposium on Computational Intelligence and Games*, pages 9–16, 2010.

[17] O. Vaněk, M. Jakob, O. Hrstka, and M. Pěchouček. Using multi-agent simulation to improve the security of maritime transit. In *12th International Workshop on Multi-Agent-Based Simulation*, pages 12–23, 2011.

[18] O. Vaněk, M. Jakob, V. Lisý, B. Bošanský, and M. Pěchouček. Iterative game-theoretic route selection for hostile area transit and patrolling. In *10th International Conference on Autonomous Agents and Multiagent Systems*, pages 1273–1274, 2011.

Improving Building Energy Efficiency with a Network of Sensing, Learning and Prediction Agents

Sunil Mamidi
Infomation Sciences Institute
University of Southern California
Marina del Rey, CA 90292
mamidi@usc.edu

Yu-Han Chang
Infomation Sciences Institute
University of Southern California
Marina del Rey, CA 90292
ychang@isi.edu

Rajiv Maheswaran
Infomation Sciences Institute
University of Southern California
Marina del Rey, CA 90292
maheswar@isi.edu

ABSTRACT

Nearly 20% of total energy consumption in the United States is accounted for in heating, ventilation, and air conditioning (HVAC) systems. Smart sensing and adaptive energy management agents can greatly decrease the energy usage of HVAC systems in many building applications, for example by enabling the operator to shut off HVAC to unoccupied rooms. We implement a multi-modal sensor agent that is non-intrusive and low-cost, combining information such as motion detection, CO2 reading, sound level, ambient light, and door state sensing. We show that in our live testbed at the USC campus, these sensor agents can be used to accurately estimate the number of occupants in each room using machine learning techniques, and that these techniques can also be applied to predict future occupancy by creating agent models of the occupants. These predictions will be used by control agents to enable the HVAC system increase its efficiency by continuously adapting to occupancy forecasts of each room.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI

General Terms

Design

Keywords

Occupancy prediction, Energy efficiency, Environmental sensing, Adaptive-agents

1. INTRODUCTION

Adaptive multi-agent systems are a key component of efforts towards reducing energy consumption, with proposed applications to smart grid and residential HVAC system operation. In this paper, we describe a multi-agent system deployed in a large educational/commercial office building environment that optimizes energy use and occupant comfort. Such a system can significantly reduce energy consumption without decreasing occupant comfort and satisfaction by adding spatiotemporal constraints that limit energy use to zones and intervals where occupants are predicted to be present. These policies are learned by observing patterns of occupant behavior and optimizing HVAC operation in response to the learned occupant models. The techniques described have wide applicability across commercial and residential building environments.

Buildings consume about 40% of all energy used in the United States, divided nearly equally between the residential and commercial sectors, with a significant portion devoted to heating, ventilation, and air conditioning (HVAC) systems [1]. While the HVAC mechanical units themselves have increased in efficiency over the years, there have not been advances in terms of using intelligent agents to improve efficiency. Intelligent agents that robustly learn and adapt to the environments in which they are deployed have the potential to greatly reduce energy consumption by proactively adjusting building HVAC systems to respond to occupant needs along multiple objectives such as minimizing energy while maximizing occupant comfort and satisfaction.

Recently the agents community has begun to develop techniques for energy efficient practices within smart grid and some building domains, primarily residential buildings [9, 10, 12, 7]. Here we focus on HVAC control in commercial buildings, though the techniques should be directly applicable in residential settings as well. The innovative application described is a Building-Level Energy Management Systems (BLEMS) project that is deploying a multi-agent system with 58 multi-modal sensors, multiple learning agents that collaboratively learn and adapt to specific occupant needs, and 74 actuators that correspond to the building's HVAC zones and the two central air handling units (AHUs). The system is shown in Figure 1. Sensor Agents in each room read environmental variables such as temperature, CO2, and sound level every minute, and record these values to a Timeline. Occupancy Estimation Agents (OEAs) use these readings to estimate the number of occupants in each room. The Policy Agent may use these estimates for reactive actions when needed, but primarily these estimates are then used by Occupancy Prediction Agents (OPAs) to predict the number of occupant who *will* be in each room in the next hour. The Policy Agent uses these predictions to adjust the heating or air conditioning actuators so that the respective rooms are warmed or cooled to the desired temperature before the occupants arrive, or to shut down the system when occupants leave. It does this by communicating with the Honey-

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

well Enterprise Buildings Integrator (EBI) system used by University to remotely control HVAC operation in many of campus buildings.

The core component of the system is the adaptive Estimation and Prediction agents that observe multiple sensors and learn patterns of occupant behavior. By modeling occupancy patterns, the BLEMS Policy agent can conserve energy by constraining heating and cooling policies to be active only during the hours when occupants are actually in specific rooms and zones of the building. We show that our Estimation Agent can achieve 95% accuracy in the occupancy estimation task, with RMSE of occupant numbers of 0.7. Prediction of the occupancy status of a room is also an important component of the system because the thermal mass of each zone sometimes requires a substantial start-up time to heat or cool the space to a comfortable temperature. We thus need to know whether a room will be in use up to an hour in advance of actual occupancy. This type of information can be inferred or predicted from learned patterns of occupant behavior. We show that we can achieve nearly 90% accuracy in this occupancy prediction task.

2. DEPLOYMENT ENVIRONMENT

The BLEMS system is currently being deployed to a 3-story research and teaching building at the *University (name withheld)*, henceforth referred to as University, as part of a program funded by the U.S. Department of Energy. The building contains lecture halls, classrooms, conference rooms, lounge spaces, and staff and student offices. The wide range of space uses enables us to test the capabilities of the BLEMS agents within different regimes, suggesting applicability to both commercial office buildings and residential spaces, as well as challenging environments such as intermittently used conference rooms. Users of the building include permanent occupants such as professors, administrative staff, and graduate research assistants, as well as temporary occupants including students attending classes and visitors at meetings. The building, along with a floorplan of the first floor showing sensor locations, in shown in Figure 2.

To test the BLEMS system prior to full deployment at University, we deployed two identical BLEMS agents within two different lab spaces at the University, which we'll refer to as Lab1 and Lab2. The labs are located in two different buildings, and are used by 4-10 students on an intermittent basis. The spaces were chosen because they represent the most challenging environment for the BLEMS agents, where multiple students share the space and have individually variable schedules. Sometimes Lab1 is completely empty for the entire day, whereas other days there are as many as 10 people in the space. Figure 7 shows probability of atleast one occupant on a typical Business day in Lab1. The BLEMS Occupancy Estimation Agent and Occupancy Prediction Agent learn the behavior patterns of these users over the course of a month of observation, and use this learned behavior to adjust heating and air conditioning policies based on the OEA and OPA estimates and predictions.

3. RELATED WORK

There is an expanding literature on agent-based HVAC control and occupant behavior modeling techniques for reducing energy consumption in residential and commercial building settings. Most of the agent literature on efficient HVAC control centers on residential settings where occupancy is considerably easier to model, and where HVAC systems are also much simpler, or on smart grid related technology [9, 10, 12, 7]. In the commercial office settings described in this paper, occupant schedules are much more variable, with professors often traveling, teaching, or attending meetings elsewhere, and where large inflows and outflows of students into classroom spaces occurs regularly. Furthermore, the HVAC system has many more interlinked controls, including central air handling units that deliver cooled air throughout the building ductwork, and individual airflow control and heating units in each zone/room. This makes the BLEMS Policy Agent's task more complex.

Occupant behavior models have also been explored by many researchers in civil and industrial engineering. The closest in spirit to the current work is a model developed by Page et al. [6], which models occupancy using a Markov chain. They develop a time series model of an occupant in particular zones of the building. This model was shown to simulate occupant behavior well in the aggregate, for example producing PDFs of arrival times that matches the actual distribution relatively well. However, the model does not attempt to predict actual occupancy on any given day and time, rather it only produces a probability density for occupancy at that day and time. Furthermore, it does not attempt to estimate the number of occupants in zones where more than one person may be present, e.g., labs or conference rooms.

In general, other work on occupancy models suffer from the same drawback. The outputs of these models tend to be probability densities, rather than specific predictions. As we will show in this paper, in many cases we can achieve higher accuracy by using other input features to a machine learning-based predictor, instead of simply counting and using the historical probabilities, or using survey data. The reason much of this work differs in spirit from our current paper is a difference in goals: for the related work, the occupancy models were used to create simulations of overall building occupancy, from which engineers could calculate a building's thermal loads and thus correctly size and provision an HVAC system. Use of these simulations in the operation of the HVAC system would typically be relegated to computing a reasonable start and end time for the HVAC system to be turned on. In marked contrast, our goal is to dynamically operate the HVAC system on a zone by zone basis, with potentially different behavior for each zone and each day and time. We actively use the occupancy models we develop to operate the HVAC system. Thus, our models cannot simply produce an aggregate probability density; instead we need accurate estimates of occupancy for every day, time, and zone.

For completeness, we survey some of this related work: [5, 11] attempted to create statistical occupancy time-series model based on occupancy survey of the people on a regular day. Richardson et al. [5] generated realistic occupancy using this model.The generated occupancy is binary information on a 10-minute resolution, which is similar to our study of prediction accuracy analysis where we have surveyed occupants of a building and generated data using probabilistic selection of occupant's typical schedules. Liao et al. [8] developed an agent-based model to simulate the occupant behavior and developed a graphical model on the probabilistic factors that effect agent behavior. Their experiment was

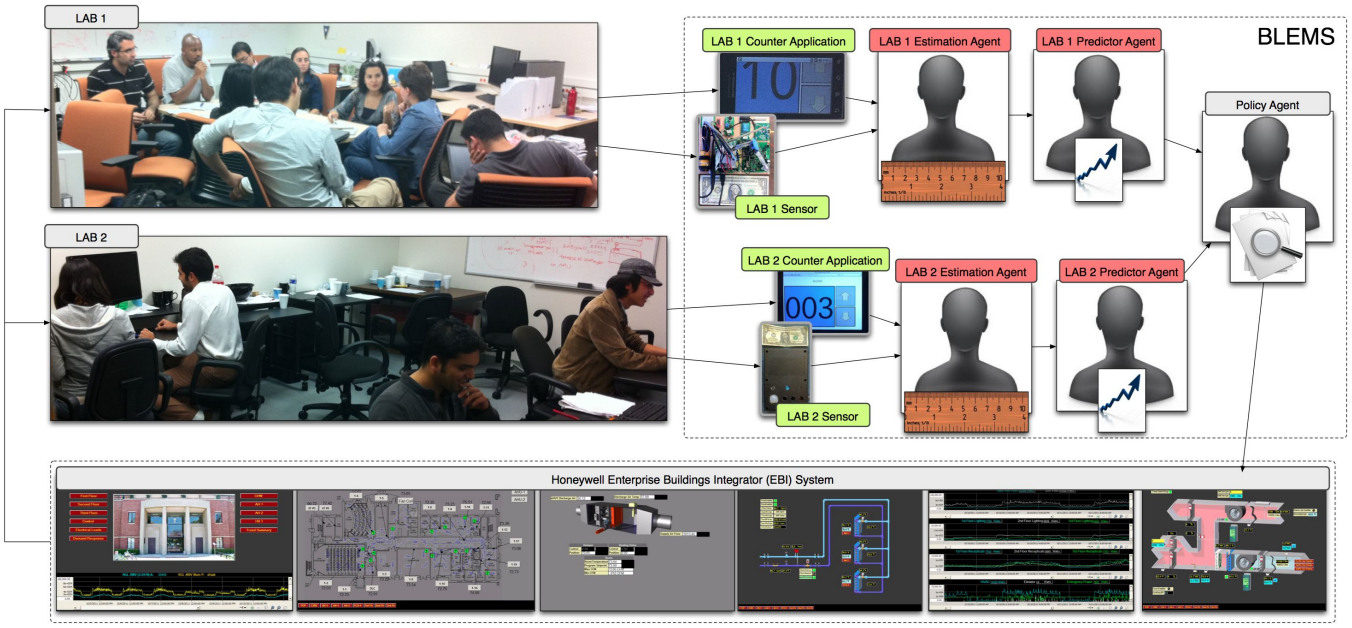


Figure 1: The BLEMS System: Sensor, Estimator, Predictor, and Policy Agents, and the Honeywell EBI.

Figure 2: (Left) The BLEMS testbed: University Building. (Right) Floorplan of first floor deployment.

limited to one occupant of a particular room.The probabilistic graphical model alone cannot predict occupancy due to dynamic nature of occupants day-to-day activity. Yu. [14] has applied a rule-based technique on motion sensor data and achieved an accuracy of 83%; they learned the rules with statistical methods in the context of single occupant in a room. In contrast to much of this work, we are building predictive models that can be deployed to a variety of offices, labs, and classrooms throughout campus buildings, and is adaptive enough to quickly learn individual occupant behaviors when deployed in the field.

Lighting is another important, though less significant, component of building energy usage. Controls based on occupancy estimates were shown to result in energy saving of 40% when the control system was able to substitute daylighting in place of artificial lighting [3]. Other types of HVAC systems, such as TABS (Thermally Activated Building System), which uses heated or cooled water circulating through pipes embedded in the floor instead of forced air, have also been investigated [4]. These investigations have used simple probabilistic occupancy models that assume arrival and departure times distributed according to Gaussian or uniform distributions, with a probabilistic rate of temporary absence. Such models are useful for evaluating traditional static policies for building operation. However, as described above, HVAC operations can be significantly optimized by responding to individual occupancy patterns rather than treating the population as homogenous.

4. SENSING AGENT

In contrast to other attempts to estimate current room occupancy, we use non-intrusive techniques that do not rely on the video or camera feeds used in prior, related work [2, 13]. Currently the most reliable estimates are based on image recognition techniques. Instead we introduce a multi-modal sensor that is low-cost and non-intrusive. Unlike the ubiquitous motion sensors deployed in "green" buildings today, a multi-modal sensor provides multiple types of readings from which we can more accurately gauge occupancy, including estimating the *number* of occupants in a room. Each modal-

ity is incorporated using fairly low-cost, off-the-shelf components. The device has the following raw sensors: sound, wide-field motion detection, narrow-field motion detection, ambient light, temperature, humidity, carbon dioxide, and door state (open/closed).

The Sensing Agent reads the values of these sensors every minute and records a useful transformation of this data onto the BLEMS Timeline. For example, it records onto the Timeline the number of times that motion was detected, rather than the raw value which is a lifetime count of motion activations. For our experiments, the Sensing Agent also retrieves ground truth occupancy counts from a Counter App that is deployed on iPads installed next to the doorways in Lab1 and Lab2. The students in these labs record their arrivals and departures using this app. This enables us to verify the accuracy of our Estimation and Prediction agents. In the future, the Sensing Agent may also receive other inputs, such as feedback from occupants using provided smartphone applications.

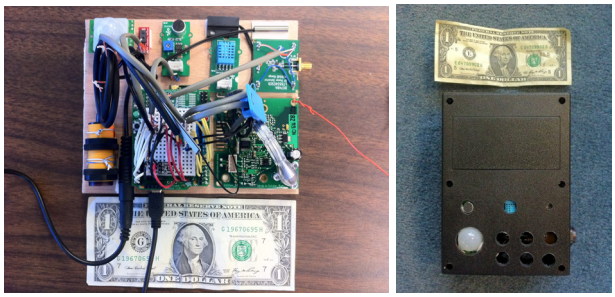


Figure 3: Prototype BLEMS sensor, with and without cover; dollar bill for scale.

5. OCCUPANCY ESTIMATION AGENT

The BLEMS system relies on accurate occupancy estimation (current number of occupants in a room) and occupancy prediction (a prediction of how many occupants will be in the room in the next 15, 30, 45, 60 minutes) in order to adjust the operation of the HVAC system to conserve energy while maintaining occupant comfort. We investigate two estimation problems: 1) estimation of whether or not there are *any* occupants in a room, and 2) estimation of the exact number of occupants in a room. The first problem, binary estimation, is clearly simpler, and we demonstrate high accuracy for that task. Solving this problem allows us to modify HVAC operation so that it is turned off when there are no occupants. The second problem is much harder, given the fairly crude sensors we are given and the goal of estimation an exact number of occupants. However, we also demonstrate surprisingly good accuracy for this task as well. Solving this problem allows us to further tune HVAC operation so that space conditioning energy (flow rate of conditioned air into the zone) is adjusted to match the number of occupants in the space, which increases comfort.

5.1 Baseline: Rule-Based Heuristic

We first implement a simple heuristic that serves as a baseline comparison for the binary occupancy estimation problem. The rule-based estimator takes as input the previous 15-minute interval of sensor data and outputs whether any occupant is present in the room. Presence is output as long as the narrow or wide field motion detector detects motion, or if the sound or CO2 sensor reads higher than the baseline normal.

5.2 Machine Learning Methods

Supervised statistical machine learning techniques use a set of labeled training data to learn the parameters of a prediction model. Here, our training set consists of the feature vectors formed from the sensor readings, where each vector is labeled with the ground truth occupancy. We then use a variety of statistical learning techniques like linear regression, logistic regression, multi-layer perceptron, and support vector machines (SVM) to train prediction models using sensor data labeled with the ground truth data. Given a new feature vector of sensor readings, the trained models can then estimate the occupancy.

It is important to note that the choice of features in the representation of the data often makes a big difference in the accuracy of the trained classifiers, depending on the type of classifier used. As described earlier, the BLEMS Sensor Agent creates a set of features that are based on the original raw sensor readings, but transformed and projected onto useful axes such as the number of times motion was detected in the last minute. The Estimation Agent adds additional knowledge to this feature vector, such as domain knowledge that biases the classification or collaborative knowledge from other agents operating in nearby or similar rooms. This overall set of features includes:

- Time: the time is the minute count from the start of the day,
- Biasing Time: in some experiments we also provide a nonlinear function that encodes the notion that occupants are more likely to be in the room during usual work hours.
- Sound: cumulative sound energy sensed for one minute,
- CO_2 : instantaneous reading of the carbon dioxide sensor,
- Number of times wide-field motion detected in the last minute, where the sensor is mounted to detect motion within the room,
- Number of times narrow-field motion detected in the last minute, where the sensor is mounted to point across the doorway,
- Temperature: Instantaneous temperature of the room recorded by sensor,
- Humidity H: Instantaneous humidity recorded by sensor,
- Motion M : Number of times motion detected by the wide beam motion detector in the last minute,
- Motion N : Number of times motion detected by the narrow beam motion detector in the last minute,
- Motion status M_0: Current wide beam motion sensor status { High=1, Low=0 },
- Motion status N_0: Current narrow beam motion sensor status { High=1, Low=0 },
- $\overline{CO_2(t_1, t_2)}$: Average CO_2 during a window of time from t_1 to t_2 hours in the past,
- $\overline{CO_2(4am, 7am)}$: Average CO_2 during 4am-7am, when occupancy is presumed to be zero,
- $\overline{O(t_1, t_2)}$: Average estimated occupancy count during a window of time from t_1 to t_2 hours in the past,
- $corr(CO_2(t_1, t_2), CO_2(t_3, t_4))$: Correlation of $CO_2(t_1, t_2)$

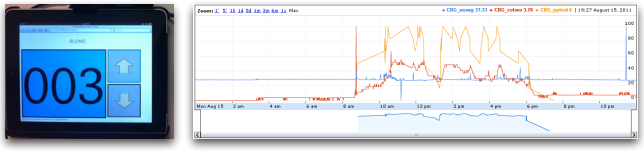


Figure 4: (Left) iPad mounted beside the lab entrance to gather ground truth occupancy counts. Large touch buttons enable occupants and visitors to easily mark their entrances and exits. (Right) A web application enables quick visualization of the sensor readings and ground truth.

and $CO_2(t_3, t_4)$, where $CO_2(t_i, t_j)$ is the vector of CO_2 per-minute readings during a window of time from t_i to t_j hours in the past,

The classifiers are trained using various subsets of this collection of features. We present results using three different subsets of features:

(i) Time, Sound, CO_2, cumulative motion count difference, cumulative beam count difference, temperature, humidity, and motion sensors,

(ii) All features in Set (i), plus $\overline{CO_2(0,3)}$, $\overline{CO_2(3,6)}$, and $\overline{CO_2(6,9)}$,

(iii) All features in Set (ii), plus $\overline{CO_2(0.5,2.5)} - \overline{CO_2(0,2)}$, $\overline{CO_2(1,3)} - \overline{CO_2(0.5,2.5)}$, $\overline{CO_2(1.5,3.5)} - \overline{CO_2(1,3)}$, $\overline{O(0,2)}$, $\overline{O(0.5,2.5)}$, $\overline{O(1,3)}$, $\overline{O(1.5,3.5)}$, $corr(CO_2(0,2), CO_2(0.5,2.5))$ $corr(CO_2(0.5,2.5), CO_2(1,3))$, and $corr(CO_2(1,3), CO_2(1.5,3.5))$.

Using these features, we estimate the occupancy count $\widehat{O(t)}$ at the current time t. We will omit the notation t when it is clear.

5.3 Experiments and Results

For the results reported in this paper, sensor devices were deployed at Lab 1 and Lab2. Both of these office spaces are shared by multiple graduate students, and the number of occupants ranges from zero to ten, with large variability within and between days. Training data was collected over several weeks. The models were then trained on this dataset. For binary occupancy estimation, we report accuracy of the predictions. For estimation of a numeric occupancy value, we report the Root Mean Square Error (RMSE). We report the average RMSE obtained through 10-fold cross validation, where in each of 10 runs, one-tenth of the training dataset is held out of the training and used as the test set.

To collect the ground truth data (the number of people who are actually in the room at that time), we mounted an iPad with a Counter App next to the doorway of the test lab (see Figure 4). Lab 1 was also outfitted with a camera that snapped an image of the entire lab every minute. Using these images, we verified the accuracy of the data collected by the Counter App, to ensure that students were using the App on a consistent basis. Our results showed that the Counter App data was a reasonable reflection of ground truth.

Rule-based heuristic. Somewhat surprisingly, the rule-based heuristic resulted in very poor results for the simple binary occupancy estimation problem. The heuristic resulted in the wrong answer more often that the correct answer; essentially the opposite of the prediction would have resulted in higher accuracies. This is due to several limitations in the rule-based heuristic. The rules are very sensitive to background fluctuations in average CO2 and sound levels. The rules are also likely to over-estimate occupancy because satisfying any one of the rules will cause the heuristic to predict that there is an occupant in the room.

Learning techniques. The feature sets used to train the occupancy estimators can greatly affect the resulting accuracy. One of the novel aspects of our learning methods is the design of the feature set. To overcome variability in certain environment variables such as CO_2, we constructed features that attempt to measure the change in background CO_2 levels throughout the day. These background changes are often due to the influence of occupants in other rooms of the building, since air is partially recirculated. The correlation features and average CO_2 features enable the classifiers to partially account for these influences. Eventually, as sensors are deployed throughout the University Building, we will be able to use communication between the agents to directly correct for some of these variations.

The core learning algorithms are primarily WEKA (open source machine learning package) implementations of standard machine learning algorithms. We report results using MultiLayer Perceptron, Linear Regression, Gaussian processes, and SVM to estimate the occupants using data from the Sensor Agent. We briefly describe each of these methods here, and provide the parameters used in each case. We did not use cross-validation to optimize the choice of parameters yet; this may be done in future work.

MultiLayer Perceptron learning has one linear node in first layer and four nodes with sigmoid activation functions in second layer. The parameters are: Learning rate 0.3, Momentum 0.2, epochs 500, error threshold 20, and one hidden layer with 4 nodes. The model denoted MLP10 is the MultiLayer Perceptron trained on feature set (ii).

Gaussian Processes learning has RBF kernel and noise of 1.0 . It was computationally expensive due the high number of matrix inverse calculations and is very time consuming for training even with a few thousand data points.

Linear Regression uses a ridge regularizer= $1.0e-8$, m5 attribute selection. The trained model estimates the number of occupants based on a linear combination of the input feature values.

SVM Multiclass classifier was also used, and the parameters of ν-SVM are $v = 0.001$, $\epsilon = 0.01$, kernel=radial basis function, cost=1.0. The model denoted SVM15 is this SVM trained on feature set (iii).

Results. Table 1 shows the accuracy and RMSE of the different estimation techniques on cross-validated training data. We show accuracy and RMSE under the two different subsets of features described earlier. For real-valued estimators, an instance is considered to be correctly classified when the estimated value is greater than 0.7 and the ground truth people count is greater than or equal to 1, or if the estimated value is less than or equal to 0.7 and the ground truth is zero.

The average RMSE for estimation with most of these techniques is less than one, which is quite good. The MultiLayer Perceptron achieves an RMSE of 0.82 on the unseen test data from the following week. An RMSE of 0.82 is a good result since the number of occupants varies between zero and ten. It suggests that our occupancy estimate is usually within one of the correct number of occupants. Given that

we are using fairly simple and crude sensors, and we have not optimized the learning process extensively, we believe this is an encouraging result.

Moreover, we used an ensemble learning method to combine results across multiple time periods. This method used a voting method to elicit the most popular prediction in the previous fifteen minutes, and used this value as its prediction. In practice, this enabled the occupancy estimator to smooth out occasional irregularities in the data and resulting predictions, leading to considerably better RMSE scores, as shown in Table 1.

Estimation method	RMSE	Accuracy
Rule-based heuristic	–	46%
MultiLayer Perceptron, featureset(i)	0.9	90%
Gaussian Processes, featureset(i)	1.0	91%
Linear Regression, featureset(i)	1.2	86%
ν-SVM-R, featureset(iii)	0.88	92%
MultiLayer Perceptron, featureset(iii)	0.73	95%
Linear Regression, featureset(ii)	1.05	87%
Ensemble Voting, featureset(iii)	0.6	95%

Table 1: Accuracy of different occupancy estimation techniques. The Ensemble Method has the best accuracy and lowest RMSE.

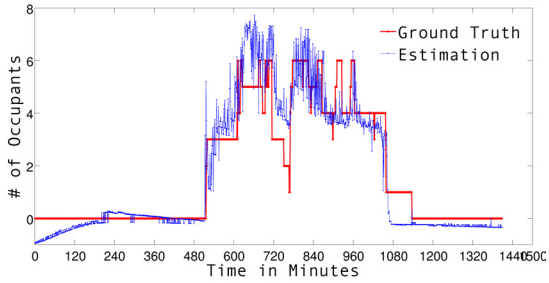
The experiments suggest that CO_2 is highly correlated to the number of occupants. Motion and motion count also correlate to presence of an occupant in room. For example, the Linear Regression learns the following coefficients for estimating the current number of occupants:

$$\begin{aligned}\widehat{O} &= -8.889 + 0.3883 * M - 0.1826 * N \\ &\quad +41.777 * CO_2 + 0.0096 * H + 0.8754 * M_0\end{aligned}$$

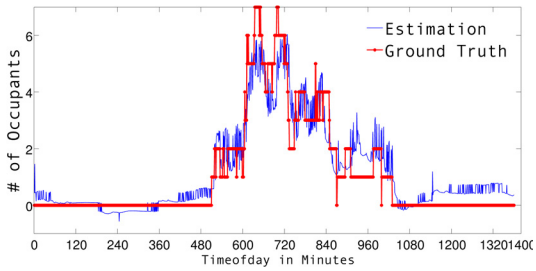
However, it is also clear that this simple classifier, while decent, does not achieve optimal occupancy estimation performance.

To get a better sense of the estimates produced throughout each day, Figure 5 shows plots for the estimated occupancy on a particular day of test data using the different occupancy estimation algorithms. Figure 9 is plot of RMSE of estimation average over a day against different dates. We can see from for Lab1 that SVM15 has low RMSE compared to MLP10. SVM15 includes autocorrelation features, which seems to improve estimation for lab which has CO_2 well correlated to Occupancy.

We also evaluate the performance of cross lab estimation: that is, using a occupancy model trained from one lab's data to estimate occupancy at the other lab. We observed an RMSE of around 2.5-3.5 for estimating Lab 2 occupancy using the Lab1 model, and an RMSE of 1.2-1.6 for estimating Lab1 occupancy using the Lab 2 model. Figure 9 shows occupancy estimation for Lab 2 using a trained model of Lab 1.



(a) MultiLayer Perceptron



(b) SVM with 15 attributes

Figure 5: Occupancy estimation using machine learning techniques.

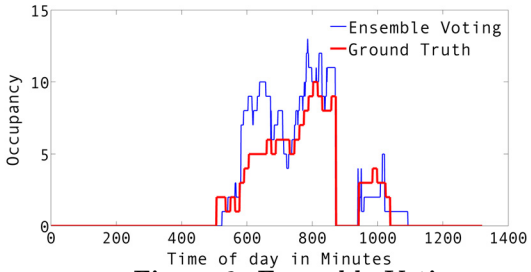


Figure 6: Ensemble Voting .

6. OCCUPANCY PREDICTION AGENT

The previous section shows that we can use BLEMS Sensor Agents to accurately estimate the number of occupants in a shared office space. On its own, this could enable significant gains in energy efficiency by enabling the HVAC system to be quickly adjusted to meet the needs of the current number of occupants. However, if we can predict the future occupancy, efficiency can be increased further. Partly, this is due to the need for unoccupied spaces to be conditioned to within a fairly tight range of temperature, so that a new occupant is not subjected to uncomfortable conditions while the space is brought to an acceptable temperature. Thus, energy is wasted maintaining all spaces within a building to within a few degrees of desired temperature. Accurate prediction of future occupancy would enable the HVAC software to completely turn off heat or air conditioning to un-used spaces. The HVAC can be turned on if occupancy is predicted far enough in advance, so that the system has ample time to prepare the room for occupancy by heating or cooling it as needed. Typically offices and shared lab spaces can be conditioned within one hour, so in this paper, we investigate the use of machine learning techniques to predict the future occupancy of building spaces for up to that interval.

As in the occupancy estimation problem described ear-

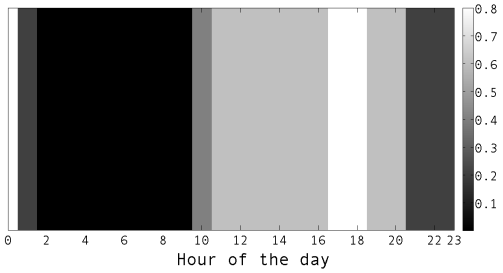


Figure 7: Probability of Occupancy status on a typical Business day over 24hr period. Higher the probability, Lighter is the heatmap.

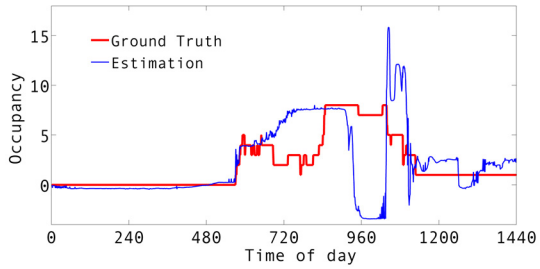


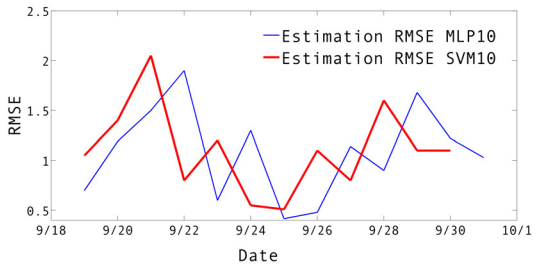
Figure 8: Cross lab testing: Estimating Lab 2 occupancy using model trained on Lab 1 data.

lier, we train the occupancy models using a labeled training dataset. Each day is divided into a feature vector of length 96, where the room's occupancy within each 15-minute interval in the day is represented by one binary feature. We train a separate model to estimate the future occupancy in the room at each 15-minute interval of the day. That is, if it is currently 11:45, to predict the occupancy at twelve noon, we train a model using a training set that has feature vectors describing the occupancy pattern from midnight to 11:45, labeled by the occupancy at noon.

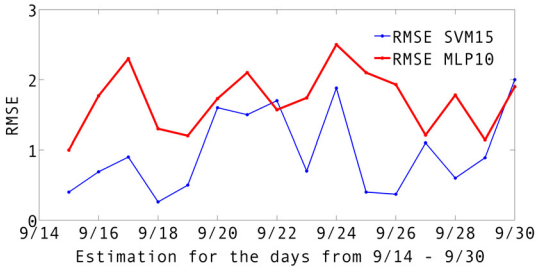
We use two different datasets for this portion of the work. The primary dataset is the same as in the occupancy estimation work, consisting of sensor data and ground truth from the deployed Sensor Agents in the two campus lab spaces described earlier. We predict future occupancy in two different scenarios: (1) assuming we only have access to the estimated occupancy counts outputted by the Occupancy Estimation Agent, and (2) assuming we have access to the ground truth.

We tested occupancy prediction using ground truth data and sensor occupancy estimation. We have used algorithms as in second dataset (described in following paragraph), except with less training data (15 to 18 days instead of 100+ days) and test data of less than a week. Table 2 shows the accuracy of prediction of estimated occupancy. The accuracy is 0.95 for occupancy prediction using ground truth data and drops to .89 for occupancy prediction using estimated occupancy.

The second dataset is derived from survey data gathered from the University Building occupants. We conducted a survey of the building occupants using a web application that asks for their three most typical schedules during the week. Based on the survey of 30 respondents, we generated simulated data using probabilistic selection of schedules with some noise added. Each of the three schedules is selected with a probability corresponding to the occupant's



(a) Lab 2 Occupancy Estimation RMSE from 9/18-9/30 by MLP10 and SVM10



(b) Lab 1 Occupancy Estimation RMSE from 9/14-9/30 by SVM15 and MLP10

Figure 9: Estimation RMSE for Lab 1 and Lab 2.

	Training Data	Test Data	Accuracy
Lab1 ,Ground Truth	15	5	0.945
Lab2, Ground Truth	18	6	0.93
Lab1, Estimated Data	15	5	0.89
Lab2, Estimated Data	18	6	0.8

Table 2: Occupancy Prediction accuracy (15 min in advance), using real data from live deployment at Lab 1 and Lab 2.

survey response. The occupancy pattern is then perturbed by changing the occupancy bit of each 15-minute interval with 0.2 probability. We use this simulated dataset to investigate the feasibility of predicting occupancy of a room up to 1.5 hours in advance.

The learning methods are trained on different combinations of size of training dataset {100,200} days, and predict the occupancy {15, 30, 60, 90} minutes in advance. The results are shown in Table 3 and Figure 10. We used a multilayer perceptron and logisitic regression classifier. We note that the best possible accuracy is 0.8, since we generated the data with a noise term of 0.2. The table shows that both methods are able to fairly accurately predict occupancy 15 minutes in advance. Prediction of occupancy 30, 60, and 90 minutes in advance is somewhat lower, but is still quite high relative to the absolute maximum of 80% accuracy. With smaller amounts of noise in the generated data, the accuracy is significantly higher, but this shows that the methods will still perform reasonably well with high degrees of noise. On the synthetic survey data, it is interesting to note that the system's performance is actually not as good. Partly we believe this is because occupant schedule are actually not as variable as the data we synthetically generated. We purposefully chose a high noise term of 0.2 in order to produce a challenging dataset. However, our live data shows that

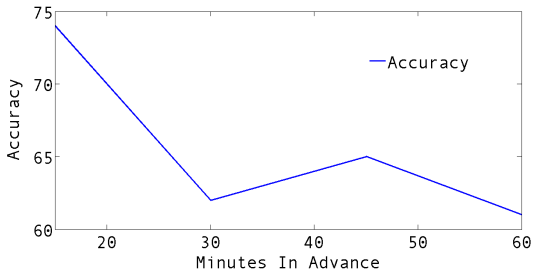


Figure 10: Occupancy Prediction: accuracy vs. time period for advance prediction.

this may have been overly pessimistic. Table 3 shows our accuracy using the survey-based synthetic data, Figure 10 shows how the accuracy degrades as we attempt to predict further into the future (up to an hour in advance).

Prediction method	Training size	Min in advance	Accuracy
Multi-layer Perceptron	100	15	67%
MultiLayer Perceptron	200	15	68%
Logistic Regression	100	15	72%
Logistic Regression	200	15	75%

Table 3: Accuracy of different occupancy prediction techniques for predicting future occupancy 15 minutes in advance, given different amounts of training data(Synthetic data generated from survey of University Building occupants).

7. CONCLUSION

Adaptive multi-agent systems that learn about occupant behaviors and optimize HVAC operation in response to these occupant models promise to greatly reduce energy consumption.We show that machine learning techniques can be used to estimate room occupancy using a set of simple sensors, and that we can use similar techniques to learn agent models that predict occupant behavior. By using these agent models to predict room occupancy up to an hour in advance, the BLEMS system can intelligently control the multi-agent HVAC system to minimize energy usage while maintaining occupant comfort.

We will continue to refine the learning methods. In particular, the current off-the-shelf methods will need to be refined to better handle small training dataset sizes (so that we can predict occupancy without lengthy collection of occupant behavior) and take advantage of additional structure in the data (such as a sequence of beam activation and motion activation indicating occupant arrival). Even with the current methods, it appears that we can handle relatively small dataset sizes of a couple weeks.

The good performance of the system on the live testbed environments enables us to proceed with the project. The BLEMS system is currently being deployed to an entire three-storey office building on the University campus. Experiments in the near future will meter the energy consumption at University Building under control conditions and under the treatment condition with the BLEMS system. We will establish the energy reduction made possible by intelligent sensing, agent modeling, and adaptive control strategies.

Acknowledgements. This work was funded in part by a grant from the U.S. Department of Energy, DE-EE0004019.

8. REFERENCES

[1] *Buildings Energy Data Book*. U.S. Department of Energy, 2010.

[2] Y. Benezeth, H. Laurent, B. Emile, and C. Rosenberger. Towards a sensor for detecting human presence and characterizing activity. *Energy and Buildings*, 43:305–314, 2011.

[3] C. R. D. Bourgeois and I. Macdonald. Adding advanced behavioural models in whole building energy simulation-a study on the total energy impact of manual and automated lighting control. *Elsevier. Energy and Buildings*, 38:814–823, 2006.

[4] W. P. Dirk Saelens and R. Baetens. Energy and comfort performance of thermally activated building systems including occupant behavior. *Elsevier. Energy and Buildings*, 46:835–848, 2011.

[5] M. T. Ian Richardson and D. Infield. A high-resolution domestic building occupancy model for energy demand simulations. *Elsevier. Energy and Buildings*, 40:1560–1566, 2008.

[6] N. M. J. Page, D. Robinson and J. L. Scartezzini. A generalised stochastic model for the simulation of occupant presence. *Elsevier. Energy and Buildings*, 40:83–98, 2008.

[7] S. Kamboj, W. Kempton, and K. S. Decker. Deploying power grid-integrated electric vehicles as a multi-agent system. In *Autonomous Agents and Multi-Agent System (AAMAS)*, 2011.

[8] C. Liao and P. Barooah. An integrated approach to occupancy modeling and estimation in commercial buildings. *American Control Conference*, 2010.

[9] Z. Mo and A. Mahdavi. An agent-based simulation-assisted approach to bi-lateral building systems control. In *IBPSA*, 2003.

[10] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. Agent-based control for decentralized demand side management in the smart grid. In *Autonomous Agents and Multi-Agent System (AAMAS)*, 2011.

[11] A. T. Rhys Goldstein and A. Khan. Schedule-calibrated occupant behavior simulation. *Autodesk Research.*, 2010.

[12] A. Rogers, S. Maleki, S. Ghosh, and N. Jennings. Adaptive home heating control through gaussian process prediction and mathematical programming. In *International Workshop on Agent Technology for Energy Systems (ATES)*, 2011.

[13] A. Sarkar, M. Fairchild, and C. Salvaggio. Integrated daylight harvesting and occupancy detection using digital imaging. In *Proceedings of SPIE (The International Society for Optics and Photonics)*, 2008.

[14] T. Yu. Modeling occupancy behavior for energy efficiency and occupants comfort management in intelligent buildings. *International Conference on Machine Learning and Applications*, 2010.

Session 2A
Virtual Agents

Bayesian Model of the Social Effects of Emotion in Decision-Making in Multiagent Systems

Celso M. de Melo
Institute for Creative Technologies, USC, 12015 Waterfront Drive, Building #4 Playa Vista, CA 90094-2536, USA
demelo@ict.usc.edu

Peter Carnevale
University of Southern California Marshall School of Business, Los Angeles, CA 90089-0808, USA
peter.carnevale@marshall.usc.edu

Stephen Read
University of Southern California Department of Psychology, Los Angeles, CA 90089-1061, USA
read@rcf.usc.edu

Dimitrios Antos
Harvard University, 33 Oxford st., Maxwell-Dworkin 217, Cambridge, MA 02138, USA
antos@fas.harvard.edu

Jonathan Gratch
Institute for Creative Technologies, USC, 12015 Waterfront Drive, Building #4 Playa Vista, CA 90094-2536, USA
gratch@ict.usc.edu

ABSTRACT

Research in the behavioral sciences suggests that emotion can serve important social functions and that, more than a simple manifestation of internal experience, emotion displays communicate one's beliefs, desires and intentions. In a recent study we have shown that, when engaged in the iterated prisoner's dilemma with agents that display emotion, people infer, from the emotion displays, how the agent is appraising the ongoing interaction (e.g., is the situation favorable to the agent? Does it blame me for the current state-of-affairs?). From these appraisals people, then, infer whether the agent is likely to cooperate in the future. In this paper we propose a Bayesian model that captures this social function of emotion. The model supports probabilistic predictions, from emotion displays, about how the counterpart is appraising the interaction which, in turn, lead to predictions about the counterpart's intentions. The model's parameters were learnt using data from the empirical study. Our evaluation indicated that considering emotion displays improved the model's ability to predict the counterpart's intentions, in particular, how likely it was to cooperate in a social dilemma. Using data from another empirical study where people made inferences about the counterpart's likelihood of cooperation in the absence of emotion displays, we also showed that the model could, from information about appraisals alone, make appropriate inferences about the counterpart's intentions. Overall, the paper suggests that appraisals are valuable for computational models of emotion interpretation. The relevance of these results for the design of multiagent systems where agents, human or not, can convey or recognize emotion is discussed.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence – *Intelligent Agents*; D.2.2 [**Software Engineering**]: Design Tools and Techniques – *User Interfaces*

General Terms

Design, Experimentation, Theory

Keywords

Emotion, Appraisals, Expression, Bayesian, Decision-Making

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

1. INTRODUCTION

Recent developments in the behavioral sciences have led to a revolution in the understanding of the role of emotion in cognition and social behavior. Contrary to the classical view of emotion as an obstacle to rational decision-making [1, 2], this research emphasizes the positive influence emotion can have in decision-making [3-5]. As a consequence, there has been growing interest on the impact emotions can have in multiagent systems [6] and several computational models of emotion have recently been proposed [7-10]. Following the initial focus on the *intra*personal effects of emotion [11, 12], these models also focus on the impact of emotion in the self's decision-making. However, the *inter*personal effect of emotion in decision-making is also interesting and important [13-15] – i.e., the impact of another's emotions on one's decision-making. In this paper we explore a computational model for the interpersonal effect of emotion in decision-making.

A useful framework for understanding the interpersonal effect of emotion is the theory of the social functions of emotion [16-18]. This theory emphasizes that emotional expressions are not simple manifestations of internal experience; rather, expressions are other-directed and communicate one's beliefs, desires and intentions [18-21]. Emotion displays, thus, help regulate social interaction. For instance, guilt occurs when someone transgresses an accepted social norm and serves as an apology, signaling regret, which, in turn, contributes to avoid reprisals from others [22]. To study the social functions of emotion in decision-making, de Melo et al. [23, 24] conducted a series of experiments where participants engaged in a social dilemma - the iterated prisoner's dilemma [25] - with different embodied agents. Even though following the same strategy to choose their actions, the agents showed facial displays of emotion that reflected different social value orientations (e.g., cooperative or competitive). The results indicated people's decision-making was influenced by the emotion displays and people cooperated more with agents which displays reflected a desire for cooperation (e.g., smile when mutual cooperation occurred in the game) than one which displays reflected selfish desires (e.g., a smile when the agent maximized its reward at the expense of the participant). Using the empirical data collected in these studies, de Melo et al. [26] then developed, based on maximum-likelihood estimation, a computational model for decision-making in a social dilemma that took into account the outcome of the dilemma and the emotion

display. Their results showed that this model was more accurate than a model which only took into account the dilemma's outcome.

Recently, we have extended their research with two experiments that address the *mechanism* by which emotions serve their social functions. To understand this mechanism two questions needed to be answered: What is the information conveyed by emotion displays? How is this information retrieved from the displays? To answer them, we looked at appraisal theories of emotion. In appraisal theories [27], emotion displays arise from cognitive appraisal of events with respect to the agent's goals, desires and beliefs (e.g., is this event congruent with my goals? Who is responsible for this event?). According to the pattern of appraisals that occurs, different emotions are experienced and displayed. Now, since displays reflect the agent's intentions through the appraisal process, it is also plausible to ask whether people can infer from emotion displays the agent's goals by reversing the appraisal mechanism. The question then becomes: can people retrieve information about how the sender is appraising the situation from emotion displays? To address this, in our first experiment we asked participants to imagine playing the iterated prisoner's dilemma with different embodied agents. Participants were always told the same outcome occurred but were shown videos of different emotional reactions from the agent. Participants were then asked questions about how they thought the agent was appraising the situation and how likely the agent was to cooperate in the future. The results showed that participants perceived the agent to appraise the outcome consistently with expectations from appraisal theories (e.g., when the agent showed anger after an unfavorable outcome, participants perceived the agent to appraise the outcome as obstructive to its goals and to blame the participant for it). Moreover, the results showed that appraisals statistically mediated [28] the effect of emotion displays on perception of how likely the agent was to cooperate in the future. This, thus, suggests that appraisals are a key component of the information conveyed by emotion displays. To verify that perception of appraisals influence perception of the agent's likelihood of cooperation, in our second experiment we explicitly manipulated perceptions of appraisal and measured the effect on perceptions of likelihood of cooperation. The manipulation consisted of having the agents, instead of showing facial displays of emotion, express how they were appraising the outcome through text (e.g., "I really don't like this outcome and I blame you for it"). The results showed that perceptions of appraisal influenced people's perception of how likely the agent was to cooperate in the future; moreover, when the expression of appraisals corresponded, according to predictions of appraisal theories, to the emotions displayed in the first experiment, the effects on perceptions of likelihood of cooperation where very similar across experiments. Overall, these studies suggest a causal model where emotion displays lead people to infer how the agent is appraising the outcome and that, in turn, leads people to infer how likely the agent is to cooperate in the future.

In this paper we propose a computational model that captures this appraisal-based mechanism for the interpersonal effect of emotion in decision-making. The model is useful for multi-agent systems for, at least, two reasons: (1) it can be used to design agents that convey through emotion displays appropriate information about beliefs, desires and intentions; (2) it can be used by agents to interpret how the other party, human or agent, is appraising the situation and, thus, infer its intentions. At its core, the model is about *inferring*, from emotion displays, how the counterpart appraises the situation and, from this, *inferring* the other's intentions in the social encounter. Because there is a strong inductive component in this model, we follow a Bayesian approach [29]. We considered three alternative Bayesian networks: the first considered the outcome of the dilemma only; the second considered the outcome and the emotion displayed; the third considered the outcome, emotion display and appraisals. The models' parameters were learnt from the empirical data collected in the first of the aforementioned studies. We compared models with respect to their accuracy in predicting the counterpart's likelihood of cooperation in the future. Our first hypothesis, following de Melo et al.'s [26] findings was that:

Models that considered emotion display would have better accuracy than models that did not *(H1)*

However, the focus of this paper is on showing the value of integrating (perceptions of) appraisals in a model of decision-making. One important advantage appraisals provide is a structure which is shared by several emotions. For instance, *conduciveness to goals* is an appraisal which is shared by joy and sadness [27]: an event which is conducive to someone's goals causes joy; an event which is obstructive to someone's goals causes sadness. This shared structure provides a mechanism for learning parameters and making inferences regarding emotions even in the absence of examples for that particular emotion. All that is necessary is data for the emotions with which the missing emotion shares appraisals. So, our next hypothesis was:

Models that considered appraisals would have better accuracy than models that did not, over test sets which included emotions not seen in the training set *(H2)*

Finally, there are situations where people express how they are appraising a situation without resorting to emotion expression. An obvious example is when people convey verbally their attitudes toward an event. The data collected in the second of the aforementioned studies – where people convey appraisals through text – is a case in point. This dataset could, thus, be used to test our third and final hypothesis:

Models that considered appraisals could be accurate even when no emotion was shown *(H3)*

The rest of the paper is organized as follows: Section 2 presents the data in the two empirical studies; Section 3 presents the Bayesian model alternatives; Section 4 describes three experiments which test each of the hypotheses; finally, Section 5 discusses the results and its implications.

2. EMPIRICAL DATA

2.1 Study 1

The Bayesian model presented in this paper is based on data collected in a recent empirical study. In this study, we gave participants scenarios where they imagined playing the iterated prisoner's dilemma with embodied agents that displayed emotion. The payoff matrix we used is shown in Table 1. Each scenario pertained to the first round (of a 5-round game) and corresponded to a particular outcome of the game. Participants were then shown a video of how the agent reacted to the outcome. The reaction corresponded to a facial display of emotion. The agents and

emotion displays used in the experiment are shown in Figure 1. The experiment followed a mixed design with two factors: *Outcome* (between-participants) with 4 levels (one for each outcome of the game); and, *Emotion* (repeated-measures) with 5 levels (Neutral vs. Joy vs. Anger vs. Sadness vs. Guilt). In other words, each participant only saw one outcome but, was engaged with several agents that expressed different emotions. Only certain pairings of outcome and emotion where explored: (a) in mutual cooperation (CC), we considered the neutral and joy expressions; (b) when the participant was exploited (participant cooperated and agent defected, $C_H D_A$), we considered the neutral, joy and guilt expressions; (c) when the participant exploited (participant defected and agent cooperated, $D_H C_A$), we considered the neutral, anger and sadness expressions; (d) in mutual defection (DD), we considered the neutral, joy and anger expressions. Considering only a subset of the pairings allowed us to avoid unintuitive pairings (e.g., expression of anger in mutual cooperation) and reduce overall participation time.

Table 1. The prisoner's dilemma payoff matrix

		Agent	
		Cooperates	Defects
Participant	Cooperates	Agent: 5 Participant: 5	Agent: 2 Participant: 7
	Defects	Agent: 7 Participant: 2	Agent: 4 Participant: 4

For each scenario, after watching the video of the agent's reaction, participants were asked several questions about how the agent was appraising the outcome. Questions referred to three appraisal variables: a) *conduciveness to goals*, which measures whether the event is consistent or inconsistent with the individual's goals; (b) *blameworthiness*, which measures whether the self or another agent is responsible for the event; (c) *coping potential*, which measures one's ability to deal with (or control) the consequences of an event. These variables were chosen because, even though several appraisal theories have been proposed [27, 30-33], there tends to be agreement that these are critical for the emotions considered in this study: joy occurs when the event is conducive to one's goals; anger occurs when the event is not conducive to one's goals, is caused by another agent and one has power/control over it; sadness occurs when the event is not conducive to one's goals; guilt occurs when the event is not conducive to one's goals and is caused by the self. Questions were asked on a 7-point likert scale (e.g., for conduciveness to goals, 1 meant "the outcome is not conducive at all" and 7 meant "the outcome is very conducive"). Several questions were asked for each appraisal variable [30, 31, 33] but, after averaging correlated questions, only four measures remained (on a 1 to 7 scale): conduciveness to goals, participant-blameworthiness, self-blameworthiness and coping potential. Finally, before moving to the next scenario, the participant was asked one question about the agent's likelihood of cooperation in the next round (scale: 1-"not likely to cooperate at all" to 7-"very likely to cooperate"). Overall, 405 participants were recruited for this experiment, resulting in an average of 100 per outcome.

For the purposes of learning a Bayesian model, the appraisal and likelihood of cooperation questions were converted into binary format: the feature was set to 'true' if the original classification

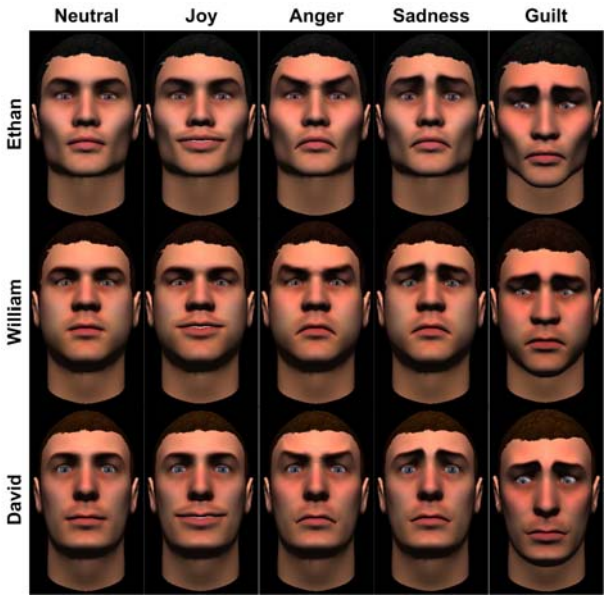


Figure 1. The facial displays of emotion.

was 5 or above; the feature was set to 'false' if the classification was 3 or below; if the classification was 4, the feature was not assigned a value (missing attribute). Each example in the training dataset, thus, had the following features:

a) Emotion Display: Neutral, Joy, Anger, Guilt or Sadness

b) Conduciveness to Goals (binary): Whether the agent was perceived to find the outcome conducive to its goals

c) Self-Blameworthiness (binary): Whether the agent was perceived to blame itself for the outcome

d) Participant-Blameworthiness (binary): Whether the agent was perceived to blame the participant for the outcome

e) Coping Potential (binary): Whether the agent was perceived to be able to deal with the consequences of the outcome

f) Likelihood of Cooperation (binary): Whether the agent was perceived to be likely to cooperate in the future

In total, excluding the examples for which the target attribute (Likelihood of Cooperation) was missing, there were 940 examples in the dataset.

2.2 Study 2

In a second empirical study, we manipulated directly how participants perceived the counterpart to be appraising the interaction, and measured perceptions of cooperation. Instead of showing emotion displays, in this study, agents expressed themselves through text in a simulated chat interface. The mapping of emotions into appraisals followed the predictions of appraisal theories [27, 30-33] and is shown in Table 2. The scenarios, game and design remained the same as in the previous study. After watching the agent's reaction, participants were asked how likely the agent was to cooperate in the next round (scale: 1-"not likely to cooperate at all" to 7-"very likely to cooperate"). Overall, 202 participants were recruited for this experiment, resulting in an average of 50 participants per outcome. The question about perception of cooperation was

discretized as in Study 1. The main difference between this and the previous dataset is that this one does not have a feature for emotion displays (or equivalently, its values are always missing). In total, the dataset had 454 examples.

Table 2. Mapping of emotion into textual expression of appraisals

Emotion	Appraisal Expression
Neutral	I neither like, nor dislike this outcome
Joy	I like this outcome
Anger	I do NOT like this outcome and I blame YOU for it
Sadness	I do NOT like this outcome
Guilt	I do NOT like this outcome and I blame MYSELF for it

3. MODELS

All Bayesian models were trained with respect to the empirical data in Study 1. Since some of the attributes in the examples could be missing (see Section 2), the EM algorithm was used for learning the parameters. The decision regarding Likelihood of Cooperation was made as follows:

- If *P(Likelihood of Cooperation) > 0.5*, true
- If *P(Likelihood of Cooperation) = 0.5*, random
- Otherwise, false

3.1 Model 1: Outcome

The first Bayesian model considered only two variables: Outcome (O) and Likelihood of Cooperation (LC). Figure 2 shows the respective Bayesian network. Outcome was set to have a uniform prior, i.e., each possible outcome occurred with 0.25 probability. The learnt parameters are shown in Table 3.

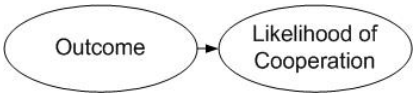


Figure 2. Bayesian network for Model 1.

Table 3. Parameters for Model 1.

O	P(LC)	O	P(LC)
CC	.470	C_HD_A	.380
DD	.405	D_HC_A	.271

3.2 Model 2: Emotion and Outcome

The second Bayesian model built on the previous and added Emotion Display (ED). Figure 3 shows the respective Bayesian network. Emotion Display was also set to have a uniform prior, i.e., each emotion occurred with 0.20 probability. The parameters are shown in Table 4.

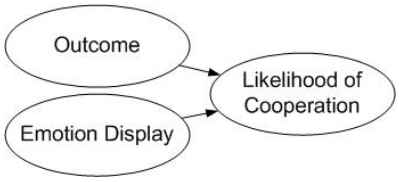


Figure 3. Bayesian network for Model 2.

Table 4. Parameters for Model 2.

ED	O	P(LC)	O	P(LC)
Neutral	CC	.235	C_HD_A	.254
Joy	CC	.719	C_HD_A	.182
Anger	CC	.500	C_HD_A	.500
Guilt	CC	.500	C_HD_A	.670
Sadness	CC	.500	C_HD_A	.500
Neutral	DD	.453	D_HC_A	.377
Joy	DD	.368	D_HC_A	.500
Anger	DD	.400	D_HC_A	.242
Guilt	DD	.500	D_HC_A	.500
Sadness	DD	.500	D_HC_A	.217

3.3 Model 3: Appraisals

The last Bayesian model added appraisal variables: Conduciveness to Goals (CG), Self-Blame (SB), Participant-Blame (PB) and Coping Potential (CP). The Bayesian network is shown in Figure 4. The appraisal variables were given BDeu priors [34], i.e., likelihood equivalent uniform Dirichlet priors. The parameters for the appraisal variables are shown in Table 5 and the parameters for Likelihood of Cooperation in Table 6.

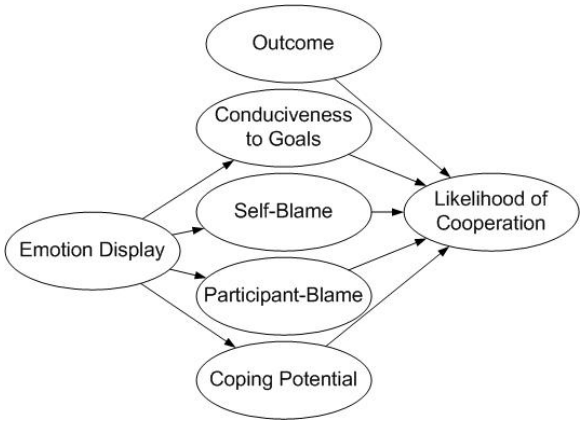


Figure 4. Bayesian network for Model 3.

Table 5. Parameters for the appraisal variables in Model 2.

ED	P(CG)	P(SB)	P(PB)	P(CP)
Neutral	.370	.203	.267	.748
Joy	.970	.206	.177	.905
Anger	.021	.381	.824	.324
Guilt	.227	.678	.222	.348
Sadness	.041	.730	.485	.285

Table 6. Likelihood of Cooperation parameters in Model 2.

CG	SB	PB	CP	O	P(LC)	O	P(LC)
T	T	T	T	CC	.436	DD	.367
F	T	T	T	CC	.082	DD	.476
T	F	T	T	CC	.410	DD	.459
F	F	T	T	CC	.129	DD	.265
T	T	F	T	CC	.837	DD	.263

F	T	F	T	CC	.002	DD	.658
T	F	F	T	CC	.640	DD	.387
F	F	F	T	CC	.324	DD	.369
T	T	T	F	CC	.146	DD	.080
F	T	T	F	CC	.259	DD	.670
T	F	T	F	CC	.054	DD	.018
F	F	T	F	CC	.172	DD	.307
T	T	F	F	CC	.990	DD	.971
F	T	F	F	CC	.014	DD	.371
T	F	F	F	CC	.776	DD	.635
F	F	F	F	CC	.203	DD	.367
T	T	T	T	C_HD_A	.320	D_HC_A	.913
F	T	T	T	C_HD_A	.849	D_HC_A	.411
T	F	T	T	C_HD_A	.084	D_HC_A	.386
F	F	T	T	C_HD_A	.528	D_HC_A	.150
T	T	F	T	C_HD_A	.108	D_HC_A	.602
F	T	F	T	C_HD_A	.863	D_HC_A	.156
T	F	F	T	C_HD_A	.243	D_HC_A	.464
F	F	F	T	C_HD_A	.526	D_HC_A	.338
T	T	T	F	C_HD_A	.502	D_HC_A	.012
F	T	T	F	C_HD_A	.366	D_HC_A	.275
T	F	T	F	C_HD_A	.335	D_HC_A	.201
F	F	T	F	C_HD_A	.383	D_HC_A	.212
T	T	F	F	C_HD_A	.642	D_HC_A	.982
F	T	F	F	C_HD_A	.821	D_HC_A	.185
T	F	F	F	C_HD_A	.122	D_HC_A	.926
F	F	F	F	C_HD_A	.398	D_HC_A	.149

4. EVALUATION

4.1 Experiment 1

To test hypothesis H1, that models which considered emotion would do better than models that did not, we tested the models accuracy with respect to the data in Study 1. Each model was re-trained using 20-fold cross-validation. The models were then compared with respect to average performance on the 20 test sets. Several standard performance measures are reported in Table 7: (a) *accuracy*, the percentage of correctly classified examples; (b) *true positives*, the number of correctly classified examples where the target (Likelihood of Cooperation) is 'true'; (c) *true negatives*, the number of correctly classified examples where the target is 'false'; (d) *false positives*, the number of incorrectly classified examples where the target is 'true'; (e) *false negatives*, the number of incorrectly classified examples where the target is 'false'. Means were compared using the 1-way independent ANOVA test.

The results showed that there was a significant difference in accuracy. In order to perform pairwise comparisons between the models, LSD post-hoc tests were performed (these are not shown in Table 7). The tests indicated that Models 2 and 3 were more accurate than Model 1. This confirmed hypothesis H1. Moreover, looking at the table, it was clear that Model 1 (based on Outcome) was making the same predictions as a game-theoretic model which always predicted defection[1]. Therefore, Outcome, by itself, seemed to be insufficient to discriminate examples in this dataset. Finally, Models 2 and 3 also seemed to be identical in their predictions. This suggested that, in this case, appraisal variables did not add more information than that provided by Emotion Display. The results also showed significant differences in the remaining variables. Looking at the true and false positive measures, it was confirmed that Model 1 always predicted defection. Still, on average, Model 1 was slightly better than Models 2 and 3, at predicting negative examples.

Table 7. Performance results for experiment 1. Means and standard deviations (in parenthesis) are shown

	acc	**tp**	**tn**	**fp**	**fn**
Model 1	62.38%	0.00	28.75	0.00	17.25
	(5.84)	(0.00)	(3.05)	(0.00)	(2.43)
Model 2	69.91%	6.15	26.05	2.70	11.10
	(7.19)	(2.08)	(3.51)	(1.66)	(3.16)
Model 3	69.91%	6.15	26.05	2.70	11.10
	(7.19)	(2.08)	(3.51)	(1.66)	(3.16)
Sig. (2-sd)	*.001**	*.000**	*.013**	*.000**	*.000**

* significant to $p<.05$

acc - accuracy; tp - true positives; tn - true negatives; fp - false positives; fn - false negatives

4.2 Experiment 2

To test hypothesis H2, that the appraisal model would have better accuracy than the others over a test set with unseen emotions, we split the data in Study 1 into two subsets: (a) the *training subset*, which included all the examples from Study 1 except the ones corresponding to Joy with the outcome C_HD_A; (b) the *test subset*, which included all the examples from Study 1 where the emotion was Joy and the outcome was C_HD_A. Models were then trained on the former subset and tested on the latter. The results are shown in Table 8.

Table 8. Performance results for experiment 2

	acc	**tp**	**tn**	**fp**	**fn**
Model 1	81.82%	0.00	72.00	0.00	16.00
Model 2	56.82%	9.00	41.00	31.00	7.00
Model 3	81.82%	0.00	72.00	0.00	16.00

acc - accuracy; tp - true positives; tn - true negatives; fp - false positives; fn - false negatives

The results showed that Model 3 was performing better than Model 2. This happened because, since there were no examples in the training set corresponding to Joy in C_HD_A, Model 2's posterior for Likelihood of Cooperation was 0.500, which corresponded to

[1] The intuition is that: the last iteration is a 1-shot prisoner's dilemma game, for which the only Nash equilibrium is mutual defection; thus, the second to last game becomes the effective last round for which a decision needs to be made. Thus, by induction, players should defect in the first round and continue doing so in every round until all rounds are completed.

a random decision. On the other hand, because of the shared appraisal structure, Model 3's posterior for Likelihood of Cooperation ($P(LC|Joy, C_HD_A)$) was 0.272. The posterior, thus, was reflecting other examples which had information about the appraisals underlying Joy. Therefore, hypothesis H2 was confirmed. Finally, the results reveal that, in this case, Model 1 performed as well as Model 3. This happened because both always defected in this test set.

4.3 Experiment 3

To test hypothesis H3, that the appraisal model could make accurate predictions even in the absence of evidence for emotion displays, we tested our models with the data from Study 2. The models were still trained on the data from Study 1 but, were tested on data from Study 2. The results are shown in Table 9.

Table 9. Performance results for experiment 3

	acc	tp	tn	fp	fn
Model 1	57.49%	0.00	261.00	0.00	193.00
Model 2	57.49%	0.00	261.00	0.00	193.00
Model 3	66.74%	72.00	231.00	30.00	121.00

acc - accuracy; tp - true positives; tn - true negatives; fp - false positives; fn - false negatives

The results showed that Model 3 was outperforming the remaining models on this dataset. This confirmed hypothesis H3. Effectively, in the absence of information about emotion displays, Model 2 could not do better than advance a prediction based only on Outcome as in Model 1.

5. DISCUSSION

This paper presents a Bayesian model that captures social effects of emotion displays in decision-making. The model's parameters were learnt using empirical data from an experiment where people engaged in a social dilemma with embodied agents that expressed emotions. The results in experiment 1 indicated that a model which took into account emotion displays was more accurate in replicating people's decision-making behavior than a model which only took into account the social dilemma outcome. This result reinforces findings in the behavioral sciences that show that non-verbal behavior – in particular, facial displays of emotion – can influence people's decision to cooperate in social dilemmas [35-38]. The results also replicate de Melo et al.'s [26] findings that a computer model of decision-making in a social dilemma improves if it takes into account the counterpart's emotion displays.

The results for Model 1, based on Outcome only and which always predicted defection, emphasize the insufficiency of a game-theoretic approach for modeling agents that interact with people. Effectively, unlike the rational prediction of defection in every round in the finite iterated prisoner's dilemma, people often cooperated in our datasets. This is compatible with the widely accepted view that people's behavior systematically deviates from game-theoretic predictions of rational behavior [39-42]. Moreover, our findings show that emotion is one of the factors that helps explain such deviations. The systematic influence of emotion displays in decision-making is, effectively, one of the premises of the social functions theory of emotion [16-18]. This theory suggests that, more than mere manifestations of internal experience, emotion expression is other-directed and communicates one's beliefs, desires and intentions. In multiagent systems research, these social effects of emotion have already been shown, for instance, when agents interact with people in social dilemmas [23, 24] and negotiation [43].

In this paper we propose further that appraisals are a useful framework to structure a computational model of emotion interpretation. Following empirical results that suggest that appraisals mediate the effect of emotion displays in decision-making, the proposed Bayesian model was structured so that variables which represented inferences about the counterpart's intentions were conditionally independent of emotion displays given information about the appraisal variables. The underlying assumption is that what matters is not the emotion display in itself but, the information it conveys about appraisals.

From a cognitive modeling perspective, it is interesting to notice that the parameters for the appraisal variables (Table 5), which represent the conditional probabilities given the emotion display, were generally in line with expectations from appraisal theories [27]: conduciveness to goals was highest for joy ($P(CG|Joy)=.970$); self-blame was highest for guilt ($P(SB|Guilt)=.678$) and sadness ($P(SB|Sadness)=.730$); participant-blame was highest for anger ($P(PB|Anger)=.824$); and, coping potential was highest for Joy ($P(CP|Joy)=.905$). This means the model was learning, from empirical data alone, some of the theoretical predictions advanced by appraisal researchers [27, 30-33].

Pragmatically, there are several advantages in following an appraisal-based model for emotion interpretation. First, appraisals provide a structure which is shared by several emotions. This provides a mechanism for learning parameters and making inferences regarding emotions even in the absence of examples for that particular emotion. The results in experiment 2 showed that the appraisal model was capable of recovering a reasonable posterior for Likelihood of Cooperation, given Joy and C_HD_A, even when no examples for that case existed in the training set. On the other hand, the model based on emotion and outcome (Model 2) could not do better than predict an even chance (0.500) of cooperation for the case where Joy is shown in C_HD_A.

A second advantage is that the appraisals model is capable of supporting inferences about the counterpart's intentions even in the absence of emotion. The results shown in experiment 3 showed that this model was capable of accurately predicting Likelihood of Cooperation for a dataset where Emotion Display was unobservable and only evidence for appraisals was available.

A third advantage of appraisals is that they provide a domain-independent mechanism for relating the counterpart's beliefs, desires and intentions to emotion displays. This relation is laid out in detail in appraisal theories of emotion [30-33] which explain how someone's beliefs, desires and intentions lead to different appraisal of situations which, in turn, lead to the experience and expression of different emotions. This knowledge can be used by multiagent system designers in, at least, two ways: (1) to implement a model, such as the one presented in this paper, that allows an agent to make inferences about the counterpart's beliefs, desires and intentions; (2) to design agents which can convey through appropriate emotions, their beliefs, desires and

intentions. Notice also that, even though appraisal theories were applied to decision-making in this paper, there is nothing in it preventing its application to other domains.

Finally, even though the paper was motivated by the literature in human-human interaction and the focus is mainly in human-agent interaction, this work has important consequences for agent-agent interaction. Simon [44] concisely articulated one of the main *intra*personal functions of emotions for intelligent agents: interrupting normal cognition when unattended goals require servicing. The theory of the social functions of emotions, on the other hand, articulates one of the main *inter*personal functions of emotions for agents: to communicate the agent's beliefs, desires and intentions. As mentioned above, appraisal theories further define how this function can be implemented through appraisals. But, why should agents use emotions to convey their mental states to other agents, as opposed to just explicitly communicate the mental states? There are many reasons, but we shall focus on two. First, from a complexity point-of-view it is more efficient for the agent to communicate information about emotions and appraisals than the whole mental state. Moreover, notice emotion need not be necessarily communicated through facial displays. Second, from an evolutionary perspective, emotion expression evolved to help solve recurrent problems that occur in social interaction [45-47]. Emotions are a quick and effective mechanism, when compared to deliberation, to respond to such problems. As multiagent systems grow in complexity, there is also an increasing need for quick and effective mechanisms to solve recurrent problems. Emotion can be one such mechanism.

6. REFERENCES

[1] Hirschman, A. 1997. The passions and the interests. Cambridge University Press.

[2] Lefford, A. 1946. The influence of emotional subject matter on logical reasoning. Journal of General Psychology 34, 127-151.

[3] Damasio, A. 1994. Descartes' error: Emotion, reason and the human brain. Putnam.

[4] Wilson, T. and Schooler, J. 1991. Thinking too much: Introspection can reduce the quality of preferences and decisions. Journal of Personality and Social Psychology 60, 181-192.

[5] Blanchette, I., Richards, A., Melnyk, L. and Lavda, A. 2007. Reasoning about emotional contents following shocking terrorist attacks: A tale of three cities. Journal of Experimental Psychology: Applied 13, 47-56.

[6] Marsella, S., Gratch, J. and Petta, P. 2010. Computational models of emotion. In A Blueprint for Affective Computing, K. Scherer, T. Banzinger and E. Roesch, Eds. Oxford University Press, Oxford, NY, 21-45.

[7] Gratch, J. and Marsella, S. 2004. A domain independent framework for modeling emotion. Journal of Cognitive Systems Research 5, 4, 269-306.

[8] Dias, J.and Paiva, A. 2005. Feeling and reasoning: A computational model for emotional agents. In Proceedings of 12th Portuguese Conference on Artificial Intelligence, EPIA 2005.

[9] Becker-Asano, C. and Wachsmuth, I. 2008. Affect simulation with primary and secondary emotions. In Proceedings of the 8th International Conference on Intelligent Virtual Agents.

[10] Wehrle, T. and Scherer, K. 2001. Toward computational modeling of appraisal theories. In Appraisal processes in emotion: Theory, methods, research, K. Scherer, A. Schorr and T. Johnstone, Eds. Oxford University Press, New York, 350-365.

[11] Loewenstein, G. and Lerner, J. 2003. The role of affect in decision making. In Handbook of Affective Sciences, R. Davidson, K. Scherer and H. Goldsmith, Eds. Oxford University Press, New York, 619-642.

[12] Blanchette, I. and Richards, A. 2010. The influence of affect on higher level cognition: A review of research on interpretation, judgment, decision making and reasoning. Cognition and Emotion 15, 1-35.

[13] Morris, M. and Keltner, D. 2000. How emotions work: An analysis of the social functions of emotional expression in negotiations. Research in Organizational Behavior 22, 1-50.

[14] Van Kleef, G., De Dreu, C., and Manstead, A. 2004. The interpersonal effects of anger and happiness in negotiations. Journal of Personality and Social Psychology 86, 57-76.

[15] Rafaeli, A. and Sutton, R. 1989. The expression of emotion in organizational life. Research in Organizational Behavior 11, 1-43.

[16] Frijda, N. and Mesquita, B. 1994. The social roles and functions of emotions. In Emotion and culture: Empirical studies of mutual influence, S. Kitayama and H. Markus, Eds. American Psychological Association, Washington, DC, 51-87.

[17] Keltner, D. and Haidt, J. 1999. Social functions of emotions at four levels of analysis. Cognition and Emotion 13, 505-521.

[18] Keltner, D. and Kring, A. 1998. Emotion, social function, and psychopathology. Review of General Psychology 2, 320-342.

[19] Bavelas, J., Black, A., Lemery C. and Mullet, J. 1986. 'I show how you feel': Motor mimicry as a communicative act. Journal of Personality and Social Psychology 50, 322-329.

[20] Fernandez-Dols, J. and Ruiz-Belda, M. 1995. Are smiles signs of happiness? Gold medal winners at the Olympic games. Journal of Personality and Social Psychology 69, 1113-1119.

[21] Kraut, R. and Johnston, R. 1979. Social and emotional messages of smiling: An ethological approach. Journal of Personality and Social Psychology 37, 1539-1533.

[22] Keltner, D. and Buswell, B. 1997. Embarrassment: Its distinct form and appeasement functions. Psychological Bulletin 122, 250-270.

[23] de Melo, C., Carnevale, P. and Gratch, J. The impact of emotion displays in embodied agents on emergence of cooperation with people. Presence: Teleoperators and Virtual Environments Journal, 2011, in press.

[24] de Melo, C., Carnevale, P. and Gratch, J. 2011. Reverse appraisal: Inferring from emotion displays who is the cooperator and the competitor in a social dilemma. In Proceedings of 33rd Annual Meeting of the Cognitive Science Society, 396-401.

[25] Poundstone, W. 1993. Prisoner's dilemma. Doubleday.

[26] de Melo, C., Carnevale, P., Antos, D. and Gratch, J. 2011. A computer model of the interpersonal effect of emotion displayed in social dilemmas. In Proceedings of the Affective Computing and Intelligent Interaction (ACII) Conference, 67-76.

[27] Ellsworth, P. and Scherer, K. 2003. Appraisal processes in emotion. In Handbook of Affective Sciences, R. Davidson, K. Scherer and H. Goldsmith, Eds. Oxford University Press, New York, 572-595.

[28] Preacher, K., & Hayes, A. 2008. Asymptotic and resampling strategies for assessing and comparing indirect effects in multiple mediator models. Behavior Research Methods 40, 879-891.

[29] Griffiths, T., Kemp, C. and Tenenbaum, J. 2008. Bayesian models of cognition. In The Cambridge handbook of computational cognitive modeling, Ron Sun, Ed. Cambridge University Press.

[30] Scherer, K. 2001. Appraisal considered as a process of multi-level sequential checking. In Appraisal processes in emotion: Theory, methods, research, K. Scherer, A. Schorr and T. Johnstone, Eds. Oxford University Press, New York, 92-120.

[31] Roseman, I. 2001. A model of appraisal in the emotion system: integrating theory, research, and applications. In Appraisal processes in emotion: Theory, methods, research, K. Scherer, A. Schorr and T. Johnstone, Eds. Oxford University Press, New York, 68-91.

[32] Ortony, A., Clore, G. and Collins, A. 1988. The cognitive structure of emotions. Cambridge University Press.

[33] Smith, C. and Ellsworth, P. 1985. Patterns of cognitive appraisal in emotion. Journal of Personality and Social Psychology 48, 813-838.

[34] Heckerman, D., Geiger, D. and Chickering, D. 1995. Learning Bayesian networks: The combination of knowledge and statistical data. Machine Learning 20, 197-243.

[35] Boone, R. and Buck, R. 2003. Emotional expressivity and trustworthiness: The role of nonverbal behavior in the evolution of cooperation. Journal of Nonverbal Behavior 27, 163-182.

[36] Frank, R. 1988. Passions within reason. Norton.

[37] Schug, J., Matsumoto, D., Horita, Y., Yamagishi, T. and Bonnet, K. 2010. Emotional expressivity as a signal of cooperation. Evolution and Human Behavior 31, 87-94.

[38] Scharlemann, J., Eckel, C., Kacelnik, A. and Wilson, R. 2001. The value of a smile: Game theory with a human face. Journal of Economic Psychology 22, 617-640.

[39] Tversky, A. and Kahneman, D. 1981. The framing of decisions and the psychology of choice. Science 211, 453-458.

[40] Simon, H. 1997. Models of bounded rationality. MIT Press.

[41] Starmer, C. 2000. Developments in non-expected utility theory: The hunt for descriptive theory of choice under risk. Journal of Economic Literature 38, 332-382.

[42] Camerer, C. 1995. Individual decision making. In Handbook of Experimental Economics, J. Kagel and A. Roth, Eds. Princeton University Press, Princeton.

[43] de Melo, C., Carnevale, P. and Gratch, J. (2011). The effect of expression of anger and happiness in computer agents on negotiations with humans. In Proceedings of Autonomous Agents and Multiagent Systems (AAMAS) 2011.

[44] Simon, H. 1967. Motivational and emotional controls of cognition. Psychological Review 74, 29-39.

[45] Darwin, C. 1872. The expression of the emotions in man and animals. Murray.

[46] Ekman, P. 1992. An argument for basic emotions. Cognition and Emotion 6, 169-200.

[47] Lazarus, R. 1991. Emotion and adaptation. Oxford University Press.

Towards building a Virtual Counselor: Modeling Nonverbal Behavior during Intimate Self-Disclosure

Sin-Hwa Kang[1], Jonathan Gratch[1], Candy Sidner[2], Ron Artstein[1], Lixing Huang[1], and Louis-Philippe Morency[1]

[1] Institute for Creative Technologies
University of Southern California
12015 Waterfront Drive
Playa Vista, CA 90094, USA
1-310-574-5700

{kang, gratch, artstein, lhuang, morency}@ict.usc.edu

[2]Dept of Computer Science
Worcester Polytechnic Institute
100 Institute Road
Worcester, MA 01609, USA
1-508-831-5000

sidner@wpi.edu

ABSTRACT

Nonverbal behavior is considered critical for indicating intimacy and is important when designing a social virtual agent such as a counselor. One key research question is how to properly express intimate self-disclosure. In this paper we present an extensive study of human nonverbal behavior during intimate self-disclosure. This is an important milestone in creating a virtual counselor. A study of video interactions between human participants demonstrated that people display more head tilts and pauses when they revealed highly intimate information about themselves; they presented more head nods and eye gazes during less intimate sharing. An implementation of these behaviors in a virtual agent suggests that people tend to perceive head tilts, pauses and gaze aversion by the agent as conveying intimate self-disclosure. These findings are important for future research with virtual counselors and other social agents.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Intelligent agents.
J.4 [**Social and Behavioral Sciences**]: Psychology.

General Terms

Experimentation, Human Factors.

Keywords

Virtual agents, Nonverbal behavior, Intimacy, Self-disclosure, Rapport, Affective behavior.

1. INTRODUCTION

Humans often share personal information with others in order to create social connections. Sharing personal information is especially important in counseling interactions [14]. Research studying the relationship between intimate self-disclosure and human behavior critically informs the development of virtual agents that create rapport with human interaction partners. One significant example of this application is using virtual agents as counselors in psychotherapeutic situations. We argue that the capability of expressing different intimacy levels is key to a successful virtual counselor to reciprocally induce disclosure in clients. Previous studies [5,6] found that human clients liked virtual counselors who disclosed personal information, only verbally. In this paper, we address the complementary challenge of learning nonverbal behavior associated to self-disclosure.

There has been substantial interest in modeling the nonverbal behavior of humans for application in developing virtual agents [4,7,8,15,16,21,24]. Patterns of nonverbal behavior have been studied in terms of a function of intimacy in social interactions [11]. Existing studies found that nonverbal behavior may indicate intimacy [2,11] by operating as a key channel for the expression of communicators' inner feelings and intentions [10,25,26]. Edinger and Patterson [11] describe that intimacy could be defined as the principal affective reaction toward the other person in interpersonal communication. Researchers further argue that nonverbal signals are more believable than verbal cues as those are impulsive and harder to be manipulated [18].

Specifically, in psychotherapeutic situations, researchers have addressed the critical role that nonverbal behavior plays in the formation and maintenance of the therapeutic relationship by shaping rapport between counselors and clients [26]. Research has found that clients' nonverbal behavior unconsciously disclosed intimate information that is not conveyed by verbal signals [14,26]. Therefore, counselors' communication with clients using nonverbal affective expression may enhance counseling effects.

Our study is an important step in building a virtual counselor. The goal of our study is to learn a model of nonverbal behavior that indicates intimacy for use by virtual counselors. Based on the literature review, our current study focuses on the investigation of humans' nonverbal behavior in association with their intimate verbal self-disclosure in interview interactions. We formulate our main research question as:

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

"*What types of nonverbal behavior does a person present when s/he discloses information with different levels of intimacy?*"

To address this research question, we analyze the data of interviewees' nonverbal behavior in conjunction with their intimate self-disclosure. This dataset was collected by Kang and Gratch in the context of a study focusing on verbal self-disclosure for virtual counselor [17]. The dataset did not have an interviewer's self-disclosure but an interviewee's self-disclosure in counseling interactions. Since nonverbal behaviors form part of general human interaction patterns, we assume that the data learned from client behavior will be useful for modeling the non-verbal behavior of counselors. We focus on six nonverbal behaviors: eye gazes, head nods, head shakes, head tilts, pauses (silence) and smiles. The choice of six nonverbal behaviors was motivated by a literature review and features diagnostic of social effects in prior work [3,9,12,18,19,23,25], as well as a pre-analysis by an expert in nonverbal communication. These six nonverbal behaviors were identified as being easily recognizable with current visions system and having the most potential.

Our results show that interesting nonverbal patterns are often associated with self-disclosure, both with individual features (e.g. head nods or head tilts) and co-occurrence (e.g. head tilts and pauses). We also present an inter-coder reliability designed for continuous behavior annotations. Based on these findings, we present a preliminary study analyzing the effect of nonverbal behavior with a virtual counselor.

The following section describes the original dataset of computer-mediated one-on-one human interviews analyzing self-disclosure. This description is necessary to understand our novel analysis described in Section 3, focusing on nonverbal behavior. In section 4, we present a preliminary evaluation by applying our findings to a virtual agent. Section 5 presents discussion and conclusions.

2. SELF-DISCLOSURE DATASET

We describe in this section the details of the self-disclosure dataset used to analyze nonverbal patterns. The video sequences analyzed in our paper were recorded during the Kang & Gratch [17] study. These interviews were computer mediated (through a video conference system). In the interview interaction, the interviewee was asked to answer ten questions asked by an interviewer that required gradually increasing levels of intimate self-disclosure. We utilize the dataset[1] of the study [17] collected in the form of computer-mediated one-on-one human and interview interaction. Since interactions with virtual agents always happen through a media (e.g., computer screen), the use of computer mediation between the two humans is motivated by creating a situation that is similar to what a human experiences with a virtual agent.

[1] The human-human dataset used in this paper was part of a more extensive design involving three conditions [17].

2.1 Participants and Procedure

Thirty-six people (50% women, 50% men) from the general Los Angeles area participated in this study. They were recruited using Craigslist.com and were compensated $30 for seventy-five minutes of their participation. On average, the participants were 36 years old (M = 36.03, SD = 8.96).

The paired participants (a confederate and a subject) never met each other beforehand. The interaction took place in two separate rooms where the paired participants were placed at different times, to avoid any initial face-to-face contact. The confederate was placed in one of the rooms before a recruited subject arrived to participate in the study. Recruited subjects were given a conversational scenario where the interviewer asked ten questions requiring gradually increasing intimacy levels of self-disclosure from the interviewee. The authors of the study [17] proposed that this communication situation and the questions could motivate emotional interaction where people need to disclose personal information about themselves to get to know each other. Interviewees and interviewers in actual interactions saw each other's video image displayed on a 30-inch computer monitor (see Figure 1). Confederates played the role of an interviewer. The typical conversation was allowed to last about thirty minutes, but interviewees were not informed of any specific time limitation. The condition of the study was presented to same gender combinations of dyadic partners: male-male and female-female.

To allow video interview conversation, video conference software (Skype) and a web-cam (Logitech QuickCam Orbit MP) were used. A hands-free headset connected to the computer was provided to both of the interviewers and interviewees for the audio communication.

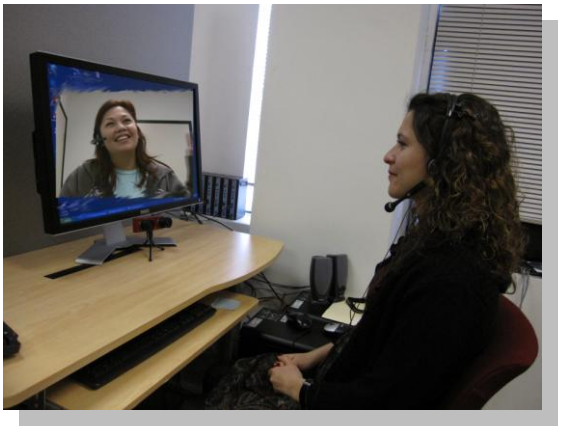


Figure 1. Computer-mediated interview interaction between an interviewer (a confederate) and an interviewee (a subject)

2.2 Measurement: Intimacy of Interviewees' Self-Disclosure

In the study [17], interviewers (confederates) did not talk about themselves, thus we decided to analyze interviewees' self-disclosure instead. The intimacy of interviewees' self-

disclosure was independently rated by two coders. The coders rated verbal data of interviewees' answers from annotated audio transcriptions.

First, the two coders defined utterances as "disclosure" or "other." The utterance is "an idea unit, which is an expression of one whole idea or proposition [17]."

Second, the coders rated the intimacy level of each "disclosure" utterance using the three layer categorization scheme of Altman and Taylor's three-layer categorization scheme [1]: a peripheral layer (lower intimacy), an intermediate layer (intermediate intimacy), and core layer (higher intimacy). The examples of each category included: "I am 30-years old (peripheral layer)" "I like to go shopping (intermediate layer)" and "I feel most guilty about cheating on my girlfriend (core layer)."

After the utterances were defined as self-disclosure and intimacy levels were judged, inter-coder reliability was measured. The authors performed Krippendorff's alpha for interval data obtained by rating intimacy levels [20]. The results of Krippendorff's alpha showed good inter-coder reliability: Alpha = .84; Do (Observed Disagreement) = 232.37; De (Expected Disagreement) = 1483.14.

Therefore, interviewees' verbal self-disclosure had four intimacy levels: 0 – no intimacy, 1 – lower intimacy, 2 – intermediate intimacy, and 3 – higher intimacy. This rating scheme was used for our main data analysis described in the next section.

3. STUDY OF HUMAN BEHAVIOR: Experiments and results

Our main goal of this paper is to find what types of interviewees' nonverbal behavior is associated with different intimacy levels of verbal self-disclosure. In this section, we first describe the types of nonverbal behavior that we explored, present the inter-coder reliability of our experiments and finally discuss our findings.

3.1 Nonverbal Features

We annotated interviewees' nonverbal behavior based on the video recordings. The nonverbal features included six types of behaviors that we considered most representative features for indicating intimate self-disclosure. Details about annotating the six nonverbal features are below:

- *Eye Gazes*: Eye gazes start when an interviewee starts to look at an interviewer and ends when he or she averts his or her gaze. An annotator looked at the full sequence and identified the gaze direction associated with the interviewer, then performed the full annotation.
- *Head Nods*: Head nods are head rotations along the vertical axis (pitch angle). A head nod gesture starts when the head moves and ends when either the head stops moving or the head nod amplitude starts increasing again.
- *Head Shakes*: Head shakes are head rotations along the horizontal axis (yaw angle). They were annotated using the same approach as head nods.
- *Head Tilts*: Head tilts are head rotations within the plane defined by the torso (head rotation around the nose).
- *Pauses*: Pauses (silence) were extracted from the audio transcription files.
- *Smiles*: Smiles were annotated using the same procedure as head nods. If a smile was slowly decreasing in amplitude and suddenly increased, we annotated it as a new smile.

While nonverbal behavior was annotated for both answers and questions, the analysis presented in this paper focuses on answers annotations. We keep as future work the analysis of nonverbal behavior during questions.

We define two types of features for each annotated nonverbal behavior:

- *Normalized Duration*: Percentage of the time the nonverbal behavior was active during the answer;
- *Normalized Count*: Number of times a nonverbal behavior occurs divided by the length of the answer (in seconds).

We normalized the duration and count to remove any confounding effect caused by a big difference of the total lengths between interviewees' answers.
The annotation work was done using the ELAN software (version 3.9.0). We assigned one coder to annotate each feature, while assigning two coders for head nods and smiles as these were considered having substantial variation among coders based on the outcomes of our previous experiment.

The outcome of the inter-coder reliability analysis on head nods and smiles is presented in the next section.

3.2 Inter-Coder Reliability of Continuous Nonverbal Behavior Annotation

We calculate reliability between two coders on the annotation of head nods and smiles. Our calculation is not concerned with the individuation of gestures (for example whether a certain time span contains one or two head nods), but only with whether the annotators agree that at a certain time point a head nod occurred. We therefore treat the annotations as an aggregation of individual time slices, and check agreement on each slice separately. The raw annotations are already digitized by the maximal resolution of the annotation tool, which is 1 millisecond; for efficient computation we only look at 50 millisecond time slices -- the difference in reliability is negligible, since head nods and smiles typically last for much longer than 50ms. Observed agreement between the annotators was 95% for head nods and 84% for smiles. That is, at 95% and 84% of the time points, annotators agreed on whether or not a head nod or smile was present.

We correct for chance agreements between the annotators using Krippendorff's alpha [20], which removes the amount of agreement expected by chance. Chance-corrected agreement is 60% on head nods and 66% on smiles, showing a good amount of agreement (the figure is lower for head nods because they occur less frequently, so a higher amount of agreement is accountable by chance; overall head nods are marked 7% of the time, whereas smiles are marked 38% of the time).

There is substantial variation in the reliability of annotation for the different experiment participants. For head nods, observed agreement ranges from 86% to 99.8% (median 95%, mean 95%, s.d. 3.5%), and chance-corrected agreement ranges from -16% to 99% (median 66%, mean 62%, s.d. 25%). For smiles, observed agreement ranges from 61% to 99.1% (median 84%, mean 84%, s.d. 9.7%), and chance-corrected agreement ranges from 17% to 98% (median 67%, mean 65%, s.d. 21%). We interpret this variation as an indication that both smiles and head nods are harder to detect on some people than others. Chance correction for individual participants was always performed using the expected agreement derived from the pooled annotation data, because the larger number of observations is likely to yield a better estimate of the true amount of agreement expected by chance.

3.3 Intimacy Levels of Interviewees' Self-Disclosure

The association between interviewees' answer intimacy and their nonverbal behavior was analyzed by categorizing three levels of intimacy[2]: Low Intimacy (N = 92), Medium Intimacy (N = 91), and High Intimacy (N = 177). The Low Intimacy included "no intimacy (0)" and "lower intimacy (1)." The Medium Intimacy included "intermediate intimacy (2)." The High Intimacy included "higher intimacy (3)."

3.4 Results of Single Feature Analysis

We ran One-Way ANOVA to find the pattern of six nonverbal behaviors associated with three intimacy levels of self-disclosure: eye gazes, head nods, head shakes, head tilts, pauses and smiles.

The results showed that there was a significant difference for *Head Nods* in Normalized Duration [$F(2,357) = 3.216$; $p = .041$; $\eta^2 = .018$] for Low Intimacy ($M = .088$, $SD = .167$) and High Intimacy ($M = .049$, $SD = .105$). The results also showed a significant difference for *Head Tilts* in Normalized Duration [$F(2,357) = 3.569$; $p = .029$; $\eta^2 = .020$] for Low Intimacy ($M = .039$, $SD = .062$) and High Intimacy ($M = .076$, $SD = .126$), as well as in Normalized Count [$F(2,357) = 4.465$; $p = .012$; $\eta^2 = .024$] for Low Intimacy ($M = .045$, $SD = .072$) and High Intimacy ($M = .080$, $SD = .122$).

The results did not show statistically significant difference for other nonverbal features.

[2] The "N" indicates a total number of subjects' answers that falls into each of three categories: Low, Medium, and High intimacy.

The analysis results are presented in Table 1 and Figure 2.

3.5 Results of Co-occurrence Analysis

We are interested not only in individual features related with intimacy but also co-occurrence patterns: when two nonverbal behaviors occur at the same time. We encode these co-occurrence features the same way as individual features:

- Normalized duration: percentage of the time both features were active
- Normalized count: number of times both features were active divided by the answer length.

The results showed that there was a significant difference for *Head Nods & Eye Gazes* in Normalized Count [$F(2,357) = 3.187$; $p = .042$; $\eta^2 = .018$] for Low Intimacy ($M = .089$, $SD = .207$) and High Intimacy ($M = .042$, $SD = .106$). The results also showed a significant difference for *Head Tilts & Pauses* in Normalized Count [$F(2,357) = 4.229$; $p = .015$; $\eta^2 = .023$] for Low Intimacy ($M = .024$, $SD = .043$) and High Intimacy ($M = .058$, $SD = .120$). The results further demonstrated that there was a moderate difference for *Head Nods & Eye Gazes* in Normalized Duration [$F(2,357) = 3.007$; $p = .051$; $\eta^2 = .017$] for Low Intimacy ($M = .054$, $SD = .121$) and High Intimacy ($M = .025$, $SD = .070$).

The results did not show statistically significant difference for other nonverbal features.

Table 1. One-Way ANOVA for single features. Our analysis shows significant differences for head nods and head tilts.

	Normalized Duration				**Normalized Count**			
	Intimacy			***p***	**Intimacy**			***p***
	Low	**Medium**	**High**		**Low**	**Medium**	**High**	
Eye Gazes	.476	.415	.408	.186	.438	.338	.352	.118
Head Nods	**.088***	.051	**.049***	**.041**	.099	.074	.057	.111
Head Shakes	.066	.092	.092	.362	.038	.055	.059	.291
Head Tilts	**.039***	.054	**.076***	**.029**	**.045***	.052	**.080***	**.012**
Pauses	.462	.469	.486	.484	.733	.682	.647	.322
Smiles	.233	.300	.271	.358	.162	.167	.159	.974

*The mean difference is statistically significant by Bonferroni Test [28]

The analysis results are presented in Table 2 and Figure 3.

There was no statistically significant difference in the patterns of nonverbal behaviors associated with different intimacy levels between males and females.

Table 2. One-Way ANOVA for co-occurrence features. Our analysis shows significant difference for the patterns (i) head nods and eye gazes, and (ii) head tilts and pauses.

	Normalized Duration				Normalized Count			
	Intimacy			*p*	Intimacy			*p*
	Low	Medium	High		Low	Medium	High	
Head Nods & Eye Gazes	**.054***	.040	**.025***	**.051**	**.089***	.066	**.041***	**.042**
Head Nods & Head Tilts	.003	.003	.001	.384	.006	.007	.003	.293
Head Nods & Pauses	.052	.031	.028	.200	.084	.058	.043	.120
Head Nods & Head Shakes	.000	.000	.002	.418	.001	.000	.008	.316
Head Nods & Smiles	.019	.017	.013	.770	.035	.043	.018	.328
Head Tilts & Eye Gazes	.013	.016	.025	.128	.026	.032	.048	.260
Head Tilts & Pauses	.012	.023	.031	.085	**.024***	.038	**.058***	**.015**
Head Tilts & Head Shakes	.002	.016	.016	.299	.004	.012	.016	.221
Head Tilts & Smiles	.011	.020	.021	.553	.016	.022	.029	.384
Head Shakes & Eye Gazes	.033	.040	.039	.790	.030	.046	.046	.436
Head Shakes & Pauses	.028	.028	.034	.741	.031	.044	.052	.251
Head Shakes & Smiles	.015	.044	.033	.151	.016	.030	.029	.445
Smiles & Eye Gazes	.100	.117	.106	.810	.138	.120	.118	.776
Smiles & Pauses	.065	.083	.083	.543	.119	.124	.123	.986
Pauses & Eye Gazes	.116	.108	.129	.522	.290	.241	.242	.394

*The mean difference is statistically significant by Bonferroni Test

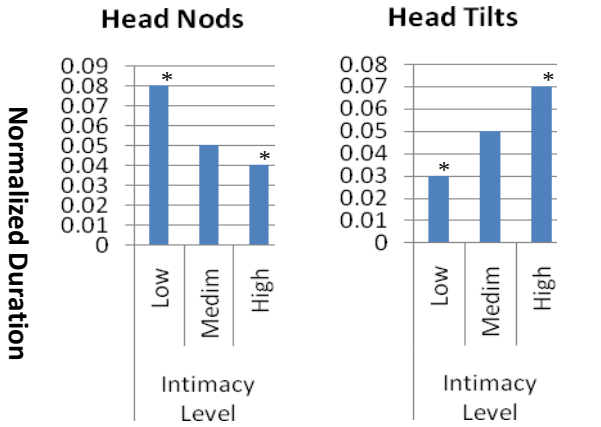


Figure 2. Mean difference of normalized duration for head nods and head tilts. We can see that head tilts are positively associated with intimacy while head nods are reduced with higher intimacy.

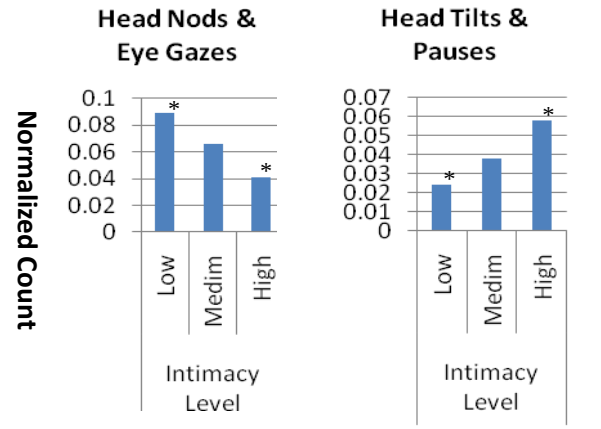


Figure 3. Mean difference of normalized count for two co-occurrence patterns. The first pattern (head nods and gazes) is inversely correlated with intimacy while the other pattern is directly associated.

3.6 Discussion

We found that interviewees showed more head tilts when they disclosed highly intimate information. Furthermore, interviewees presented more head tilts with silent pauses when they revealed highly personal information. Hesitant responses accompanied by pauses were considered unreliable reactions [11], and may mostly have been presented to avoid feelings of embarrassment that could happen when someone revealed intimate information about him or herself. These findings demonstrate that head tilts and pauses are strong nonverbal cues that convey high intimacy.

We also found that interviewees presented less head nods. Head nods are a cue of a positive response in most cultures, e.g. American culture. We contend that interviewees would hesitate to show head nods when they disclosed highly personal

information, whereas they would present more head tilts as a signal of thinking to give appropriate answers in a polite manner.

Finally, head shakes and smiles were not affected significantly by intimacy levels of interviewees' self-disclosure. Head shakes are described as a feature dependent on accompanying vocal signals that imply suspicion, dissatisfaction or impossibility, while presenting "no" without saying it [18]. Therefore, head shakes are considered a signal of negative responses in most cultures, e.g. American culture. In the type of an interview interaction utilized in this study, we argue that it would not be common for interviewees to show such a negative response during their interactions with interviewers who were supposedly strangers to the interviewee. Meanwhile, smiles can be interpreted in different ways depending on social context. Smiles are usually perceived as expressions of liking and acceptance [14,27], whereas some people use smiles to hide their anxiety in a polite way [26]. Therefore, there is no right answer to interpret any finding related to the smile feature.

There was no statistically significant difference for the gaze feature, but, in general, interviewees looked at an interviewer more when they gave less intimate answers (see Table 1 & Figure 2). A similar pattern was found in the study by Exline and his colleagues [13]. They discovered that participants showed greater gaze toward an interviewer while they gave answers responding to more innocuous questions compared to more intimate ones in an interview interaction. We, however, found that interviewees showed more eye gaze accompanying head nods while interviewees were giving less intimate information about themselves (see Table 2 & Figure 3). These outcomes imply that head nods may be a strong cue representing low intimacy in communication.

The outcomes demonstrated that interviewees displayed more head tilts and pauses when they revealed highly intimate information about themselves; they presented more head nods and eye gazes during less intimate sharing.

4. IMPLEMENTATION IN A VIRTUAL HUMAN: Preliminary evaluation

We designed a short online survey to evaluate the effect of these nonverbal cues (identified in the previous section) with a virtual counselor. We hypothesized that people would perceive that a virtual counselor disclosed highly intimate self-disclosure if the counselor presented head tilts and pauses accompanied by gaze aversion.

4.1 Online survey design

In the survey, we created a webpage which included a question, a video clip, and two options to choose (See Figure 4). In the video clip, a virtual counselor was presented. The counselor disclosed highly intimate self-disclosure while demonstrating nonverbal behavior. The virtual counselor's nonverbal behavior was composed of head tilts and pauses accompanied by gaze aversion to represent high intimacy. The gaze aversion was applied to make the counselor's nonverbal behavior represent high intimacy as we found that humans showed mutual gazes and nods more for lowly intimate self-disclosure.

The final video clip was created after removing the counselor's voice so that participants would not know which words were spoken by the counselor. The video lasted fifteen seconds. As shown in Figure 4, the participants were instructed to select the spoken text that best match the nonverbal behavior of the virtual counselor. Two options were offered: low intimacy statement (option 1) and high intimacy statement (option 2). Both options were different from the spoken words by the original video, to assure that the participants could not guess based on the lip movement. The text for both options comes from validated previous work on self-disclosure [22].

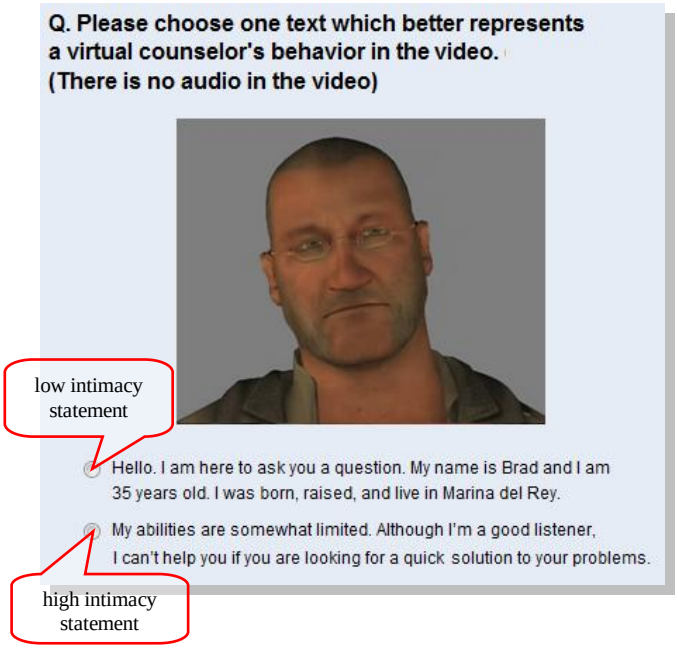


Figure 4. Web study layout. Virtual counselor displays the same nonverbal behaviors identified in our analysis (see Section 3): head tilts and pauses accompanied by gaze aversion.

4.2 Participants and Procedure

Fifteen participants (47% women, 53% men) took the survey voluntarily without any compensation. The participants were recruited via the email lists of our company and friends. On average, the participants were 27 years old (M = 27, SD = 5.0).

The participants were given the URL of the survey through an email. In the survey, participants watched a fifteen second video clip without audio and were asked to choose the spoken text that best correlates with the virtual counselor's nonverbal behavior shown in the video clip.

4.3 Results

Figure 5 shows the results of our user study. Ten participants selected the high intimacy statement while only 5 participants selected the low intimacy statement. Although still preliminary given the limited number of participants, this result is promising and gives us guidelines for the large-scale user study with a fully interactive virtual counselor.

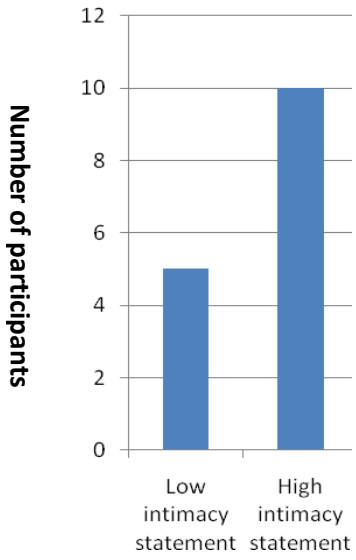


Figure 5. Results from our user study. Twice as many participants selected the high intimacy statement (option 2) over the low intimacy statement (option 1).

5. DISCUSSION AND CONCLUSIONS

Our study of nonverbal behavior in association with intimate self-disclosure provides future directions for designing virtual agents who talk about themselves during counseling interactions. Based on the outcomes of our current study, we argue that virtual counselors should show head nods and eye gazes for less intimate self-disclosure and head tilts and pauses for highly intimate self-disclosure. We contend that virtual counselors' intimate self-disclosure accompanying with appropriate nonverbal behavior will enable human clients to like their counselors more and create better rapport with them as was demonstrated by Bickmore and his colleagues [5,6] for verbal-only self-disclosure.

We presented a preliminary user study based on our findings related to intimacy and nonverbal behaviors. In the study, we focused more on finding whether users could perceive a counselor's "highly intimate" self-disclosure by looking at his/her non-verbal behaviors and associate the correct statement with the non-verbal behaviors. Our results are promising and pave the way for a large-scale user study with an interactive virtual counselor. For example, human clients will interact with the virtual counselor in counseling sessions, in which the counselor will present different types of nonverbal behavior associated with different levels of intimate self-disclosure. We also plan a future user study to look at other types of nonverbal behavior such as body movements, and the co-occurrence of more than two nonverbal behavioral features will be further investigated.

6. ACKNOWLEDGEMENTS

This study was funded by the U.S. Army Research, Development, and Engineering Command and the National Science Foundation under Grants # IIS-0916858 and IIS-0917321.The content does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

6. REFERENCES

[1] Altman, I. & Taylor, D. 1973. Social penetration: Development of interpersonal relationships. Holt McDougal.

[2] Argyle, M., & Dean, J. 1965. Eye-contact, distance, and affiliation. Sociometry, 28, pp. 289-304.

[3] Bavelas, J.B., Coates, L., Johnson, T. 2000. Listeners as co-narrators. Journal of Personality and Social Psychology 79(6), 941-952.

[4] Bee, N., Wagner, J., André, E., Vogt, T., Charles, F., Pizzi, D. & Cavazza, M. 2010. Discovering Eye Gaze Behavior during Human-Agent Conversation in an Interactive Storytelling Application. In: 12th International Conference on Multimodal Interfaces and 7th Workshop on Machine Learning for Multimodal Interaction.

[5] Bickmore, T. 2005. Ethical Issues in Using Relational Agents for Older Adults. Paper presented at the AAAI Fall Symposium on Caring Machines: AI in Eldercare, Washington, DC.

[6] Bickmore, T., Schulman, D., & Yin, L. 2009. Engagement vs. Deceit: Virtual Humans with Human Autobiographies. In: Proc. 9th International Conference on Intelligent Virtual Agents.

[7] Buschmeier, H., Bergmann, K. & Kopp, S. 2010. Adaptive Expressiveness -- Virtual Conversational Agents That Can Align to Their Interaction Partner. In: Proc. 9th International Conference on Autonomous Agents and Multiagent Systems. Toronto, Canada.

[8] Cassell, J., Pelachaud, C., Badler, N., Steedman, M., Achorn, B., Becket, T., Douville, B., Prevost, S., & Stone, M. 1994. Animated conversation: rule-based generation of facial expression, gesture & spoken intonation for multiple conversational agents. In: Proceedings of 21st Annual Conference on Computer Graphics and Interactive Technologies SIGGRAPH'94.

[9] Duncan, S. Jr. 1974. On the Structure of Speaker-Auditor Interaction during Speaking Turns. Language in Society, 3(2), pp. 161-180.

[10] Cacioppo, J. T., Petty, R. E., Losch, M. E., & Kim, H. S. 1986. Electromyographic activity over facial muscle regions can differentiate the valence and intensity of affective reactions. Journal of Personality and Social Psychology, 50, pp. 260-268.

[11] Edinger, J. & Patterson, M. 1983. Nonverbal Involvement and Social Control. Psychological Bulletin 1983, vol. 93, no. 1, pp. 30-56.

[12] Ekman, P. 1992. An argument for basic emotions. Cognition & Emotion 6(3), 169-200.

[13] Exline, R., Gray, D., & Schuette, D. 1965. Visual Behavior in a Dyad as Affected by Interview Content and

Sex of Respondent. Journal of Personality and Social Psychology 1965, vol. 1, no. 3, pp. 201-209.

[14] Farber, B. 2006. Self-Disclosure in Psychotherapy. Guilford, New York.

[15] Heylen, D. 2006. Head gestures, gaze and the principles of conversational structure. International Journal of Humanoid Robotics, 3(3), pp. 241-267.

[16] Jonsdottir, G.R., Thorisson, K.R., & Nivel, E. 2008. Learning Smooth, Human-Like Turntaking in Realtime Dialogue. In: Proceedings of International Conference on Intelligent Virtual Agent, Tokyo, Japan.

[17] Kang, S. & Gratch, J. 2010. Virtual Humans Elicit Socially Anxious Interactants' Verbal Self-Disclosure. Journal of Computer Animation and Virtual Worlds, 21(3-4), pp. 473-482.

[18] Knapp, M. & Hall, J. 2010. Nonverbal Communication in Human Interaction. Wadsworth | Cengage Learning, Boston.

[19] Kraemer N. C. 2008. Human behavior in military contexts, chap. Nonverbal Communication, pp. 150-188. Washington: The National Academies Press.

[20] Krippendorff, K. 2004. Content Analysis, an Introduction to its Methodology, 2nd Edition. Thousand Oaks, CA: Sage.

[21] Mattman, M., Gratch, J., & Marsella, S. 2005. Natural behavior of a listening agent. In: Proceedings of Interactional Conference on Intelligent Virtual Agents, Kos, Greece.

[22] Moon, Y: Intimate exchanges 2000. Using computers to elicit self- disclosure from consumers. Journal of Consumer Research, Vol. 26, No. 4, 323-339.

[23] O'Leary, M. & Gallois, C. 1985. The last ten turns: Behavior and sequencing in friends' and strangers' conversation findings. Journal of Nonverbal Behavior 9(1), Spring 1985, Human Sciences Press.

[24] Pelachaud, C. 1996. Simulation of face-to-face interaction. In: Proceedings of the workshop on Advanced Visual Interfaces, pp. 269-271, Gubbio, Italy.

[25] Philippot, P., Feldman, R., & Coats, E. 2003. The Role of Nonverbal Behavior in Clinical Settings. In: Philippot, P., Feldman, R., & Coats, E. (eds.) Nonverbal Behavior in Clinical Settings. Oxford University Press, New York.

[26] Tickle-Degnen, L. & Gavett, E. 2003. Changes in nonverbal behavior during the development of therapeutic relationships. In: Philippot, P., Feldman, R., & Coats, E. (eds.) Nonverbal Behavior in Clinical Settings. Oxford University Press, New York.

[27] Trees, A. & Manusov, V.: Managing Face Concerns in Criticism Integrating Nonverbal Behaviors as a Dimension of Politeness in Female Friendship Dyads, http://onlinelibrary.wiley.com/doi/10.1111/j.1468-2958.1998.tb00431.x/pdf.

[28] Tabachnick, B. & Fidell, L. 2001. Using Multivariate Statistics. Allyn & Bacon.

A Sequential Recommendation Approach for Interactive Personalized Story Generation

Hong Yu and Mark O. Riedl
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
{hong.yu, riedl}@cc.gatech.edu

ABSTRACT

In story-based games or other interactive story systems, a Drama Manager is an omniscient agent that acts to bring about a particular sequence of plot points for the user to experience. We present a Drama Manager that uses player modeling to personalize the user's story according to his or her storytelling preferences. In order to deliver personalized stories, a Drama Manager must make decisions on not only which plot points to be included into the unfolding story but also the optimal sequence of the events the user should experience. A prefix based collaborative filtering algorithm based on users' structural feedback is proposed to address the sequential selection problem. We demonstrate our system on a simple interactive story generation system based on choose-your-own-adventure stories to evaluate our algorithms. Results on human users and simulated users show that our Drama Manager is capable of capturing users' preference and generating personalized stories with high accuracy.

Categories and Subject Descriptors

I.2.1 [**Artificial Intelligence**]: Applications and Expert Systems—*games, Industrial automation*

General Terms

Algorithms, Design

Keywords

Interactive story generation, drama manager, player modeling, prefix based collaborative filtering

1. INTRODUCTION

Computer games use story to motivate player activity and to create a sense of causal continuity across a series of challenges [15]. While stories in games are often linear, progressively more games and virtual simulated environments allow variability in the story. A *Drama Manager* (DM) is an omniscient agent that monitors the virtual world in which the user is immersed and acts to determine what happens next in the player's story experience, often coordinating and/or instructing virtual characters [1]. In short, a Drama Manager is an agent that reasons about and attempts to enhance the user's experience in a virtual world.

Prevailing approaches to Drama Management develop agents that select successive plot points in response to player actions [9, 20, 14, 13, 7, 8]. In these systems, the DM is a surrogate for the human designer who provides some form of high-level specification for a "good" story experience. Although these action-response systems may improve player enjoyment, DM decisions do not take into account the user's preferences. We argue that the DM must also be a surrogate for the user by taking into consideration the user's preferences; the DM should model and act on the user's individualistic preferences do deliver optimized, personalized experiences.

Player modeling is a widely applied technique in computer games to capture users' preferences in order to increase enjoyment and reduce frustration and boredom [3]. Previous approaches to player modeling in story-based games have attempted to optimize player experience by classifying players according to well-defined player types [10, 4, 18]. and using pre-defined mappings of classes to plot point selection rules. These approaches require a designer to pre-determine the meaningful player types, even though there is no clear evidence of links between player types models and preferences for story content. User modeling can be cast as a recommendation process and collaborative filtering (CF) has been successfully applied to modeling user preferences over movies, products, books, music, etc [17]. Collaborative filtering algorithms attempt to detect users' rating "patterns," extract similarities between users' patterns, and make predictions of new user's ratings based on previous ratings from the users who share similar patterns. Collaborative filtering can be applied to the problem of selecting the plot point to occur next in a game, although to the best of our knowledge CF has not previously been used for Drama Management.

The techniques described above—player type classification and collaborative filtering— make one-shot recommendations and/or decisions. Unfortunately, stories are *sequences* of plot points and a player's assessment of the story he or she is experiencing is thus a function of the history of plot points experienced so far. In this paper, we present a novel approach to player modeling in games to select the most optimal sequence of plot points in a game: *Prefix Based Collaborative Filtering* (PBCF), which learns user preferences over fragments of story and then applies it to the Drama Management problem of selection of successive plot points in a game.

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

As a form of collaborative filtering, PBCF is a robust approach to player modeling in story-based computer games that uses machine learning to learn the most appropriate dimensions for the model from structured feedback—i.e., ratings of previously played story content. Player models can be built without making rigid assumptions about the model dimensions and those models will be more capable of describing, distinguishing, and capturing users' preferences. We show how a DM agent can make more effective decisions about how to act on the world for the benefit of the user, essentially optimizing the player's game experience. It is not enough to make a local decision about what should happen next in a game; a DM must determine the best possible future sequence of events the user should experience based on all previous events. Therefore, our DM agent, using PBCF, learns preferences over sequences of story events.

Our contributions are as follows: We (1) propose a new algorithm, PBCF, that performs sequential recommendation; (2) build a flexible and robust preference model that does not rely on assumptions about the model dimensions; (3) incorporate the preference model into the Drama Manager which is allowed to select plot points based on users' preferences instead of user actions.We evaluate the algorithm in a simplified testbed domain based on Choose-Your-Own-Adventure book serials[1] using human users and simulated users.

2. RELATED WORK

Drama Manager agents have been widely used to guide the users through an expected story experience set by designers. Two approaches to drama management—search-based drama management [20, 9, 16] and declarative optimization-based drama management [14, 2]—transform the problem of selecting the next best plot point into a search problem where the DM searches for possible future histories of plot points based on an evaluation function set by the designer. The Façade interactive drama [8] uses a reactive plot point selection technique to determine the next set of behaviors for two virtual characters. Riedl and colleagues [13] use a partial-order planner to re-plan a story when the user performs actions that change the virtual world in ways that prevent story progression as expected. Similarly, Porteous and Cavazza [11] use a planner with designer-provided constraints to control virtual characters and push a story forward. A DM by Magerko et al. [7] predicts player actions and attempts to prevent story failures by directing virtual characters to perform actions or change goals. These Drama Management techniques all respond to player actions to move the story forward in a way partially or completely conceived by a human designer. That is, the DM is a surrogate for the human designer.

Relatively little work has been done to determine how a story should unfold in a game or virtual environment based on player models. The PaSSAGE system [18] automatically learns a model of the player's preference through observations of the player in the virtual world, and uses the model to dynamically select the branches of a Choose-Your-Own-Adventure style story graph. PaSSAGE uses Robin's Laws five game player types: Fighters, Power Gamers, Tacticians, Storyteller, and Method Actors. A player is modeled as a vector where each dimension is the strength of one of the types. As the player performs actions, dimensions are increased or decreased in accordance to rules. Peinado and Gervás [10] use the same player types. Seif El-Nasr [5] uses a four-dimension player model: heroism, violence, self-interestedness, and cowardice. These player modeling techniques assume players can be classified according to several discrete play styles and that, even with continuous characteristic vector combining the discrete user styles, optimal story choices can be made by a DM. These systems further assume that role playing game player classifications (or ad-hoc types) are applicable to story plot choices. In addition, these systems assume that plot points could be selected in isolation from each other based on a comparison between their attributes and the player model. In this paper, we propose a collaborative filtering based player modeling approach that learns player model dimensions from user feedback—ratings—and further solves sequential plot point recommendation/selection problems.

Roberts, et al. [14, 2] developed an algorithm, Targeted Trajectory Distribution Markov Decision Process (TTD-MDP), to solve non-Markov Decision Processes by wrapping all the previous MDP states into one node of a trajectory tree. Their objective was to produce probabilistic policies for the trajectory tree that minimize divergence from a target distribution of trajectories. They apply their process to declarative optimization-based drama management by modeling stories as state space trajectories. TTD-MDPs require a target distribution across trajectories/stories. Further, as a reinforcement learning technique, it must simulate a user. While the simulated user may utilize a player model, that model would need to first be acquired. Our approach learns the player model and does not require a target distribution over trajectories.

3. PREFIX BASED COLLABORATIVE FILTERING

A story can be decomposed into a sequence of plot points, which usually represent single events or tasks in the story. In a interactive story-based game or virtual world, a Drama Manager is in charge of which plot points to be presented to the users and in what order. This is a NP complete problem given all the plot points. To address this, temporal and semantic constraints are imposed among these plot points to reduce the size of the story space [20, 9]. When constraints are known, a *branching story graph* containing the possible successors to each plot point can be built automatically or by hand. The question a DM must answer is: what is the best path in the branching story graph for an individual user? By answering the quest with a player model, we aim to create an optimal experience for the a particular user.

The architecture of our Drama Manager system is illustrated in Figure 1. The DM has an existing story library or database which contains all possible story permutations, as described in the next section. The DM obtains the current system states from the interactive system interface. Then the best path is selected by the DM according to the player model, which is built entirely based on player feedback.

3.1 Story Representation

A branching story graph is a representation that specifies which plot points are allowed to follow other plot points. For the purposes of a Drama Manager agent, it provides

[1]http://en.wikipedia.org/wiki/Choose_Your_Own_Adventure

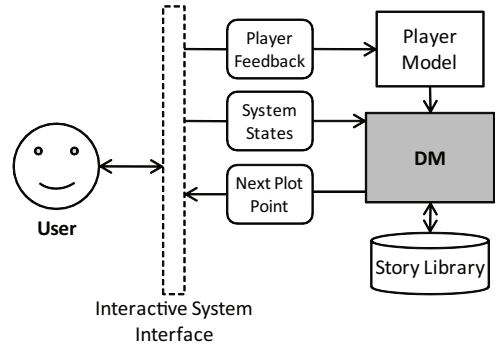


Figure 1: The architecture of the interactive story generation system.

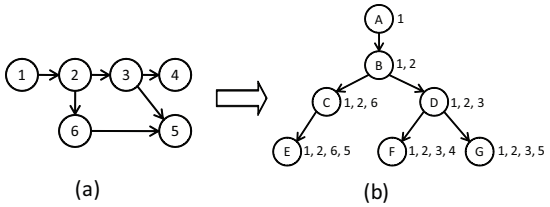


Figure 2: (a) Branching story graph of a simple story library which contains three stories. (b) The prefix graph of the story library.

the set of options for the next plot point at any given time. Figure 2(a) illustrates a very simple branching story graph, where nodes represent plot points and links represent possible alternative successors. While simple, many other plot representations are reducible to the branching story graph representation [12, 14]. A story is a path through the graph starting at the root node and terminating at a leaf node. Figure 2(a) contains three complete, possible stories ({1,2,3,4}, {1,2,3,5}, {1,2,6,5}). Note that the branching story graph is usually a graph instead of a tree.

We transform it into a prefix graph as in Figure 2(b). In the prefix graph, each node represents a prefix of a story. The children of a node are those prefixes that can directly follow the parent prefix. Apparently, the prefix graph will be a tree or forest since any prefix cannot have more than one parent node. In our system, only the prefix graph is stored in the story library.

In our approach, we ask users to rate the "story-so-far"—the portion of the story that they have observed leading up to the current point in time. Notice that it is easier and more accurate for the users to rate the story-so-far instead of each new plot point since history matters in stories. Because the branching story graph has been transformed into a prefix graph, the rating is stored with the prefix that corresponds to the story-so-far. Further, the system does not need to solve a credit assignment problem as in reinforcement learning to determine how much of a final rating each plot point is responsible for.

3.2 Player Modeling

Different users can different preference over stories. We aim to extract the dimensions of these preferences from the users' ratings instead of constraining ourselves to a few pre-defined dimensions as in prior works by Thue et al., [18], Peinado et al. [10], and [4]. The basic assumption of our player modeling algorithms is that those people who share similar preference in the past tend to share it again in the future, hence we use a form of collaborative filtering.

Prefix	User 1	User 2	User 3	...
A (1)	*	*	2	...
B (1, 2)	1	*	2	....
C (1, 2, 6)	*	*	*	...
D (1, 2, 3)	4	3	*	...
...	...	...	...	...

Figure 3: Illustration of the prefix-rating matrix. *A*, *B*, *C* and *D* represent the prefixes. The larger the digital number, the higher the preference. The stars represent those missing ratings.

Unlike in traditional recommendation systems, where collaborative filtering is usually used as one-shot recommendation of content based on preferences, we must solve the problem of sequences of recommendations where the order of plot points matters. In other words, the story generation can be viewed as as a non-Markov Decision Process, where at each step the DM's selection of optimal next plot point is based on all previous plot points. For example, if the story prefix {1, 2, 3} has been presented to the user (node D in Figure 2(b)), then the DM's next selection should be based on all the user's ratings on previous three prefix nodes (A, B, and D). A user who leaves positive feedback on node B and negative feedback on node D should be different from another one who leaves negative feedback on both node B and D. A prefix based CF approach is proposed to model players in a way that allows non-MDP problems to be solved. While other techniques similarly roll history into state nodes, as in our prefix trees and equivalent structures in TTD-MDPs [14, 2], our approach uses structured feedback to guide tree navigation, inferring ratings when feedback data is sparse.

In this paper, stories are presented to the user plot point by plot point and a preference rating for the story-so-far is collected after every plot point. Then a *prefix-rating matrix* including the story prefix ratings from all users can be obtained. A n by m prefix-rating matrix contains the ratings for n prefixes from m users. One column of the matrix represents all the ratings of the corresponding user for the all the prefixes. One row of the matrix represents ratings for the corresponding prefix from all the users. Figure 3 shows a simple illustration of the prefix-user matrix. The matrix is usually very sparse, i.e. containing a lot of missing ratings, because there is no way of expecting any given user to have read and rated all the prefixes in the library. The prefix-user matrix is treated as the product-user matrix as in traditional CF [17].

Two collaborative filtering learning algorithms are tested in this paper: *probabilistic Principle Component Analysis* (pPCA) [19] and *Non-negative Matrix Factorization* (NMF) [6, 21]. Next we briefly introduce the two algorithms and their application in our player modeling.

3.2.1 Probabilistic PCA

For a n dimensional vector $\mathbf{r}$, probabilistic PCA assumes

that it can be factorized as follows:

$$\mathbf{r} = W\mathbf{x} + \boldsymbol{\mu} + \boldsymbol{\epsilon} \quad (1)$$

where $\mathbf{x}$ is a n' dimensional vector in the hidden or reduced dimension space (usually $n' < n$) and W is a n by n' matrix. $\boldsymbol{\mu}$ is the mean vector which permits $\mathbf{r}$ to have nonzero mean. $\boldsymbol{\epsilon} \sim \sigma^2\mathrm{I}$ is the Gaussian noise.

Let the vector $\mathbf{r}$ be any one column of the prefix-rating matrix. pPCA projects the corresponding user's prefix-rating vector into the hidden space or the reduced dimension space $\mathbf{x}$ just as in traditional principle component analysis. The hidden space vector $\mathbf{x}$ models the corresponding user's preference type. Note that from Equation 1 we can get:

$$\mathbf{r}|\mathbf{x} \sim N(W\mathbf{x} + \boldsymbol{\mu}, \sigma^2\mathrm{I}) \quad (2)$$

Thus the basic assumption of pPCA algorithm is that the user's prefix rating vector (the column of the prefix-user matrix) obeys a multi-dimensional Gaussian distribution.

If the prefix-user matrix contains missing values, the EM algorithm can be used to compute W and σ [19]. The hidden space vector $\mathbf{x}$ can then be computed from the observed ratings, and the missing ratings can be estimated with W, σ and $\mathbf{x}$.

3.2.2 Non-negative Matrix Factorization

The purpose of NMF is to factorize an n by m matrix R as follows:

$$R = W * H \quad (3)$$

where $W \in \mathbb{R}^{n*m'}$ and $H \in \mathbb{R}^{m'*m}$ are two non-negative matrices (usually $m' < m$). Non-negative here means that all the entries in the matrix are greater than or equal to zero. If R contains missing values, the EM algorithms can be used to compute W and H [21]. Then the missing values in R are recovered using the estimated W and H.

If R is the prefix-rating matrix (n prefixes and m users), the m' columns of the matrix W, $\boldsymbol{w}^j$ $j = 1, ...m'$, can be viewed as a set of bases that represent different types of users. Then $\boldsymbol{h}^i$, the i^{th} column of H, will correspond to the i^{th} user's preference. In practice, it will be difficult to interpret the player types that correspond to each $\boldsymbol{h}^i$. However, if we have prior knowledge about some preference types (e.g., fighter, tactician), that is, we know their ratings for all the prefixes (e.g., fighter's rating vector $\boldsymbol{w}^f$, tactician's rating vector $\boldsymbol{w}^t$), then the matrix W can be seeded with the rating vectors ($\boldsymbol{w}^f$, $\boldsymbol{w}^t$) as fixed columns. Simulated experiments in Section 4 shows that such prior can indeed increase player modeling accuracy when they are known to accurately distinguish users with regard to stories.

3.3 Player Modeling Processes

The entire Drama Management system is composed of two phases: model training and story generation. In the model training phase, the process can be summarized as follows:

1. Build the story library storing the prefix forest.
2. Collect data and populate the prefix-rating matrix R.
3. Compute the player model parameters: W, σ and $\boldsymbol{\mu}$ for the pPCA, or W for NMF.

For a new user, after we get some initial ratings $\boldsymbol{r}$ from him or her, the story generation phase is as follows:

1. Model the new user's preference using $\boldsymbol{r}$ through computing $\boldsymbol{x}$ for pPCA, or $\boldsymbol{h}$ for NMF.
2. Calculate the full rating vector $\boldsymbol{r}'$ (with no missing values) from $\boldsymbol{x}$ using Equation 1 or Equation 3.
3. Select the highest rated full-length story that is a descendant of the current prefix in the prefix graph. Present the corresponding next plot point in the selected full-length story.
4. Collect user's rating on the story-so-far (i.e., the recommended prefix).
5. Include the new rating into $\boldsymbol{r}$ and go to step 1.

Note that it is not necessary to collect ratings after each prefix in the story generation phase; we do it in our system for the purpose of collecting as much data as possible to build a more accurate player model. With every new rating, the DM will get better understanding of the current user's preference and recommend next prefixes with higher confidence.

4. EVALUATION AND RESULTS

Based on the theory that all interactive systems can be translated into a branching story graph, we built a simple interactive storytelling system that can automatically guide the user through a particular branching path. The system presents the stories to the user one plot point by one plot point. After every plot point, it asks the user for the preference rating on the story-so-far. Instead of the user choosing the branch, the system then recommends the next branch by some means (using our model, or random). In the system, the ratings are digital integers ranging from 1 to 5. While our test bed is simple compared to modern computer games, it represents the fundamentals of other Drama Managers. By limiting player interaction to providing ratings of the story-so-far we aim to control the experimental variable of player agency to further facilitate validation of our preference model. We have performed two sets of experiments: on human users and on simulated users.

4.1 Story Library

The story library is built through transcribing the stories from four Choose-Your-Own-Adventure books—*The Abominable Snowman*, *Journey Under the Sea*, *Space and Beyond*, and *The Lost Jewels of Nabooti*—all of which are adventure stories. Every book contains a branching story tree. At the end of each page in the book, the reader is presented with a multi-choice question, the answer to which leads the reader to different pages of the book to continue down different branches of the story. Figure 4 shows the branching story graph from one of the books.

We chose to transcribe Choose-Your-Own-Adventure books to control for story quality, as opposed to authoring stories ourselves. In the system, every story is pruned and transcribed to contain exactly six plot points[2]. As in Figure 2, these branching story graphs are transformed into the prefix graphs which are stored in the story library. Thus our story library is a forest containing 154 possible stories (about 1000 words per story) and 326 prefixes.

[2]We do this for implementation purpose. It is not necessary for every story to contain exactly the same number of plot points; our system can be easily extended to handle stories of varied number of plot points.

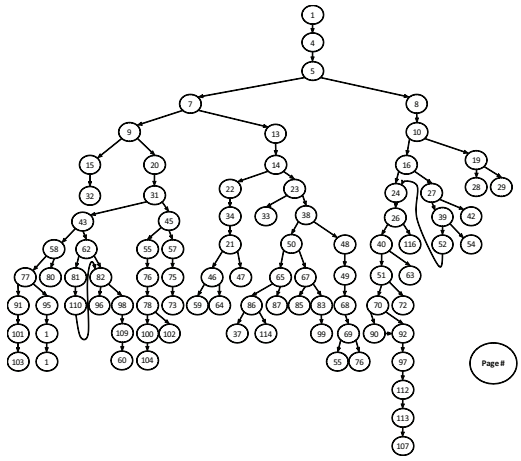


Figure 4: Illustration of the branching story representation of stories in *The Abominable Snowman*. The nodes in the graph represent pages in the book (plot points). Every story start from the root node and end on one of the leaves.

4.2 Experiments with Human Users

We conducted an evaluation of our system on human users. The evaluation consisted of two phases: model training, and story generation testing. In the model training phase, 31 users have participated in the experiments (18 male and 13 female). 26 of them are college graduate students and the other 5 users are research scientists and staff. All the users who have never read the choose-your-own-adventure stories are given a sample adventure story which is out of the story library to familiarize themselves (five out of the 31 users have been exposed to choose-your-own-adventure series before the experiments).

Every user in the training phase read ten stories randomly selected from the library. A random story is a random walk from a root of any tree in the forest to a leaf node. Each story is presented to the user one plot point at a time and a rating is collected after every plot point. The experiment took about half an hour for each player. We obtain a 326 by 31 prefix-rating matrix R with $\sim 86\%$ ratings missing.

We computed the Root Mean Square Error (RMSE) in order to get the best parameters for model training. The prefix-rating matrix R is randomly split into training part R^t which contains 90% of the ratings, and validation part R^v which contains the remaining 10% of the ratings. Note that R^t and R^v are still the same dimension as the original matrix R and both of them contains missing values. We train the NMF and pPCA on the training set R^t with different parameters. The resulting models are used to predict the ratings in the validation matrix. The Root Mean Square Error (RMSE) can be computed as follows:

$$\text{RMSE} = \sqrt{\frac{1}{|O|}\sum_{i,j\in O}(R^v_{ij} - R^{v'}_{ij})^2} \tag{4}$$

where $R^{v'}$ is the predicted validation matrix, O is the set of entries indices that are not missing in the validation matrix R^v and $|O|$ is the number of entries that are not missing in R^v. The random splitting process is repeated for ten times and the average RMSEs on the validation sets are reported in Table 1. The dim$i$ in the table mean the number of columns of the matrix W in Equation 3 is i. The RMSEs in the table suggest that there are probably six types of users in current training set when it comes to story preferences.

Algorithms	RMSE
NMF_dim3	1.2423
NMF_dim4	1.1781
NMF_dim5	1.1371
NMF_dim6	0.9901
NMF_dim7	1.1108
NMF_dim8	1.1354
NMF_dim9	1.2464
pPCA	1.2016

Table 1: The average RMSE for different parameters of NMF and pPCA algorithms. NMF_dim$i$ means NMF algorithm with the number of player styles (number of columns of the matrix W) i.

	Random	Personalized	Accuracy	p-value
All	2.9449	3.8899	0.828	< 0.0001
Existing	3.032	4.035	0.863	< 0.0224
New	2.8993	3.8138	0.809	< 0.0001

Table 2: The average ratings for the random and personalized full-length stories. The accuracies are the percent of pairs in which the average rating of the personalized stories is larger than the average rating of the random stories.

Another 22 graduate students (17 male and 5 female) were recruited for the second phase: testing of the model's ability to predict ratings. This phase is divided into four steps. In the first step, the users read five random stories and leave ratings after every plot point, as in the training phase. In the second step, the DM starts to choose new personalized stories and branches according to the users' ratings and the player models built in the training phase. The final NMF and pPCA models are trained on the entire prefix-rating matrix R with the best parameters which correspond to the least RMSE. These personalized stories are presented to the users plot point by plot point in the same way as the first five random stories and the users' ratings after every plot point are collected. Then, as in Sharma et al. [16], the DM then presents another five personalized stories in the third step, followed by five random stories in the last step in order to eliminate any prejudice introduced by the order in which the stories are presented to the users. Thus, every user in the testing phase is required to read 20 stories (10 total random stories and 10 total personalized stories).

For comparison of model performance on new users versus existing users, we also invited 11 participants from the training phase back to also participate in the validation phase. The experiment process is exactly the same as above. Table 2 shows the results for the new users and existing users when the player model is trained with NMF algorithms set for six dimensions (the variant with the lowest RMSE).

Results are shown in Table 2, the first line exhibits the statistical results on all the 32 testing users. The second line and the third line give the results of the 11 users from the

training group and the 21 users out of the training group respectively. The first column "Random" and the second column "Personalized" show the average ratings of all the random and all the personalized stories in the story generation phase respectively. For every user in the story generation phase, we also compare the pair of average story ratings from the first step and the second step, and the pair of average story ratings from the third and the fourth step. The "Accuracy" column shows the percent of pairs in which the average rating of the personalized stories is larger than the average rating of the random stories, indicating the DM correctly choosing preferred stories. The last column shows the significance of the difference between random and personalized averages using a one-tailed t-test.

4.3 Experiments with Simulated Users

In order to establish more complete analysis on PBCF, we also conducted experiments with simulated users. Simulated users are more consistent over time, allowing us to make observations about our algorithm on a controllable data set to study its capability. Note that it is not necessary for the simulated users actually being good imitations of the human users' preferences. Instead, as long as the simulated users are consistent, they can be used to experiment with the capability of our system to capture users' preference and build user models. In addition, we can get as much rating data as we need from simulated users.

The simulated users are built based on the Robin's Laws player types, which assumes there are five types of players for games: Fighters (who prefer combat), Power Gamers (who prefer gaining items and riches), Tacticians (who prefer thinking creatively), Storytellers (who prefer complex plots) and Method Actors (who prefer to take dramatic actions) [18, 10]. Every simulated user is assigned with a five-dimensional characteristic vector. Each entry of the vector (ranging from 0 to 1) specifies the corresponding characteristic of the simulated user. For example, vector $[0.8, 0, 0, 0.6, 0]$ means the simulated user is a combination of fighter and storyteller and tends to enjoy fighting a little more.

To run experiments with simulated users, all story prefixes in our database were labeled according to Robin's Laws player types. The labeling of prefixes was performed by three college students, one of whom is an author on this paper. To mitigate bias, the label for each prefix is the average label produced by each of the human labelers. Thus each prefix label is a five-dimensional vector, where each element expresses the average belief about how the story-so-far matches players of different types.

We assume that a simulated user of a particular type according to the Robin's Laws player types tends to prefer a story or story prefix that most closely matches the user's type. For example, a simulated user with characteristic vector $\mathbf{u} = [0.8, 0, 0, 0, 0]$ will prefer for a story prefix i with label $\mathbf{p}_i = [1, 0, 0, 0, 0]$ over a story prefix j with label $\mathbf{p}_j = [0, 1, 0, 0, 0]$. Consequently, we assume that the rating r of a simulated user $\mathbf{u}$ for a prefix $\mathbf{p}$ is proportional to cosine distance between the vector $\mathbf{u}$ and $\mathbf{p}$: $r \sim \frac{\mathbf{u}^T\mathbf{p}}{|\mathbf{u}||\mathbf{p}|}$. In practice, the ratings are computed by scaling the cosine distances to between 1 and 5. In addition, we add random noise with standard Gaussian distribution (mean 0 and variance 1) to all the ratings in order to simulate the human user case where it could be inaccurate for the human users to quantitate preference into digital labels.

During the model training phase of the experiments, we generate 120 simulated users with characteristic vectors randomly chosen from {[1, 0, 0, 0, 0], [0, 1, 0, 0, 0], [0, 0, 1, 0, 0], [0, 0, 0, 1, 0], [0, 0, 0, 0, 1]}. Each simulated user then reads 10 random stories and leaves a preference rating after every plot points as human users. A 326 by 120 prefix-rating matrix is populated at the end of the model training phase.

The testing phase is divided into four steps that are exactly the same as the human user's experiment story generation phase. In every step each simulated user is required to read five random or personalized stories. To test the generalization ability of our algorithm, a new group of 1000 simulated users are used in the testing phase. Each is assigned with a random characteristic vector (the five entries of the characteristic vector are values ranging from 0 to 1).

For the purpose of comparison, we implement all the following algorithms:

- *BaselineP*: pPCA is used to learn the player models from simulated users' ratings on full-length stories instead of prefixes, then directly recommends the full-length stories instead of choosing branches through recommending prefixes. This algorithm behaves similar to a traditional movie recommendation system where full-length movies are recommended based on others' ratings on the full-length movies.
- *BaselineN*: The same as *BaselineP* except using NMF.
- *Vector*: A vector based player modeling algorithm that is similar to the model learning technique used by Thue et al. [18]. A vector that simulates a player starts out as $[0, 0, 0, 0, 0]$. For every plot point encountered, the DM updates the characteristic vector based on the features of the current story prefix including the new plot point. The DM generates successive plot points by recommending the following prefix based on the updated user vector, or chooses randomly when there is no clear preference.
- *pPCA*: The prefix based algorithm using pPCA, same as with the human users.
- *NMFwoP*: The prefix based algorithm using NMF without prior knowledge, same as with the human users.
- *NMFwP*: The prefix based algorithm using NMF with Robin's Laws player types as prior knowledge as discussed in Section 3.2.2. In the case of simulated users, we can compute the correct rating vector $\boldsymbol{w}^j$ for each known player type j, where $j = 1, ...5$ correspond to the five player types in the training phase. Then these vectors $\boldsymbol{w}^j$ can be included into the matrix W in Equation 3 as fixed columns during training process.

The experiment results for these algorithms on the 1000 testing simulated users are shown in Table 3. The results are all statistically significant at p-values approaching zero (using one-tailed t-tests on random and personalized averages) due to the large number of testing users.

It is interesting to explore the learning speed of the player model as the number of stories read in every step changes, which was set to 5 for testing with human and synthetic users. Figure 5 shows the average accuracies of 1000 simulated users for different algorithms as the number of stories read in every step changes. As shown in the figure, the NMF algorithms can achieve accuracies higher than 70% even when one new simulated user reads only one story.

Algorithm	Random	Personalized	Accuracy
BaselineP	2.2190	2.5305	0.668
BaselineN	2.1752	2.4582	0.643
Vector	2.2010	2.8335	0.617
pPCA	2.2350	2.9607	0.798
NMFwoP	2.2362	3.3950	0.894
NMFwP	2.2013	4.0027	0.949

Table 3: The testing results for the simulated users using several variations of the DM algorithm.

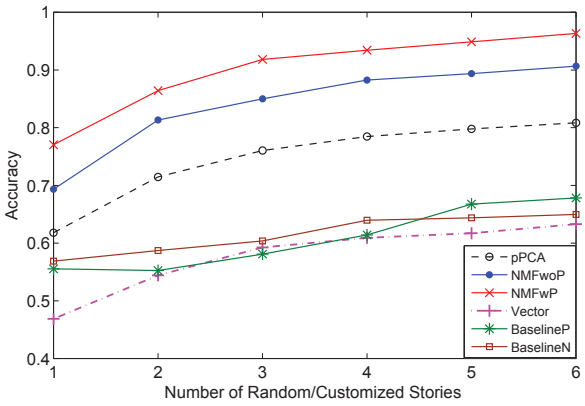


Figure 5: The accuracies of the six algorithms as the number of stories read in every step changes.

We also test the influence of the training set size on the player model training process. Figure 6 shows the average RMSEs of the three prefix based algorithms with different number of simulated users for training. Each RMSE value in the figure is an average computed from 10 random splittings of the training data. As seen from the figure, the training RMSEs decrease as the the training set size grows. Due to the Gaussian noise in the rating data, the RMSE values for the NMFwP algorithm become stable after the number of training users goes above 100 even if it has the perfect prior knowledge of the player models.

4.4 Discussions

Both the experiments on human users and simulated users achieve high story generation accuracies on the current Choose-Your-Own-Adventure data set for the prefix based algorithms. We observe that over 80% of the time, new human users will rate DM-generated stories higher than random stories. We achieve this rate after the new users have only rated 5 sample stories. The accuracy of about 86% is achieved when the testing users' data are already part of the trained model. The average ratings for the personalized stories are higher than the random stories. The results show that our prefix based player modeling algorithms can capture the users' preference and generate new stories with high confidence.

Even with Gaussian noise added to the synthetic ratings, our player modeling algorithms achieve higher accuracies on the simulated users than on the human users. Although from the learning process we know that there does exist features of story rating behavior that are predictive of future rating behavior, it is still difficult if not impossible to interpret

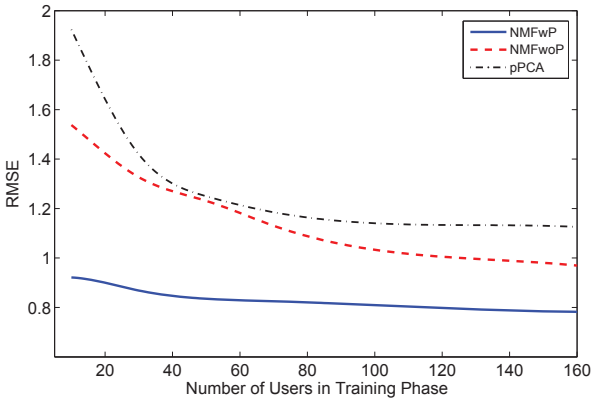


Figure 6: The average RMSEs of the three prefix based algorithms with different number of simulated users for training.

these features. We do not believe that these features are as clear cut as Robin's Laws player types, Seif El-Nasr's types, or other factorized personality models. The preference types of human users should be more complicated than the linear combinations of several presumed categories, which is also the reason to build the player models from data instead of constraining ourselves to a few pre-defined dimensions.

For the simulated users, the NMF algorithm usually performs better than the pPCA algorithm which could due to our linear model assumption for the simulated users. The linear characteristic model for simulated users coincides with the basic assumption of the NMF algorithm, which also assumes that users (columns of the matrix R in Equation 3) are linear combinations of a set of bases (columns of the matrix W in Equation 3). Although NMF is a natural fit for the synthetic users, it is also superior to pPCA for human data in terms of RMSEs in our experiments.

Figure 5 shows that the prefix based algorithms can acquire player preference much faster than applying traditional CF algorithms directly on full-length stories (baseline algorithms *BaselineP* and *BaselineN*). The main reason is that the prefix based algorithms can obtain more preference information (the ratings on all the prefixes) from users than the baseline algorithms in both model learning and story generation phases. It also demonstrates that these ratings on prefixes do strongly correlate with users' preference and can help to improve player models. The figure also shows that the Vector approach learns the player model much slower than our algorithms and is thus less accurate on average. This is because the Vector approach cannot acquire any information from the training data.

Although there are only 154 full-length stories and 326 prefixes in the story library, the well-known scalability of collaborative filtering algorithms suggests that our algorithms can be extended to handle larger scale problems as long as we have enough rating data. In traditional recommendation systems, CF algorithms can easily process products-user matrix with dimension of hundreds of thousands and achieve high recommendation accuracies [17]. The number of prefixes could be exponential in the number of plot points. But in practice we can effectively add constraints between plot points to limit the size of the prefix database. Notice that in our system, the number of total prefixes grows linearly with

the number of total stories given a limit of maximum number of plot points in each story because of the constraints imposed by the branching story graph representation.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we propose a PBCF algorithm to address the sequential recommendation problem. The player models built with the PBCF algorithm enable the Drama Manager to make successive story choices on behalf of the users. We examine the algorithm in a simple interactive story generation system built with choose-your-own-adventure series. The evaluation results on both human users and simulated users demonstrate that our algorithm is able to capture the users' preference and generate stories with high confidence.

Although the current system is simple, it represents one of the most important fundamentals of Drama Management: delivering new stories based on a pre-authored library of legal stories through player modeling. Although player modeling techniques in Drama Management has not been widely explored, we believe that it is an essential part of building an agent that is responsible for optimizing the player's experience in a game or virtual world. Our approach combines the robustness of machine learning with intuitions about the sequential nature of stories. The Drama Manager agent built in this way is capable of generating personalized stories and guiding the users through a better experience.

There are many avenues for future work. A larger story space is required to verify the scalability of our algorithms. The experiment results on simulated user show that prior knowledge does help to increase the accuracies. It will be interesting to investigate how to include prior knowledge in the case of human users. Furthermore, the player choices and other unstructured feedback are still missing in the current system. The DM operates like a story generator in current system, although we implement this as a control to validate the algorithm. When the algorithm is placed in a full virtual world that supports human player choice, then other well-known techniques for managing player choice, such as re-planning [13] can be added to the player modeling.

6. REFERENCES

[1] J. Bates. Virtual reality, art, and entertainment. *Presence: The Journal of Tele-operators and Virtual Environments*, 1(1):133–138, 1992.

[2] S. Bhat, D. L. Roberts, M. J. Nelson, C. L. Isbell, and M. Mateas. A globally optimal algorithm for TTD-MDPs. *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems*, 2007.

[3] D. Charles, M. McNeill, M. McAlister, M. Black, A. Moore, K. Stringer, J. Kĺźcklich, and A. Kerr. Player-centred game design: Player modelling and adaptive digital games. *Proceedings of DiGRA 2005 Conference*, 2005.

[4] M. S. El-Nasr. Interactive narrative architecture based on filmmaking theory. *International Journal on Intelligent Games and Simulation*, 3(1), 2004.

[5] M. S. El-Nasr. Engagement, interaction, and drama creating an engaging interactive narrative using performance arts theories. *Interactions Studies*, 8(2):209–240, 2007.

[6] D. D. Lee and H. S. Seung. Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562, 2001.

[7] B. Magerko and J. E. Laird. Mediating the tension between plot and interaction. *AAAI Workshop Series: Challenges in Game Artificial Intelligence*, 2004.

[8] M. Mateas and A. Stern. Integrating plot, character and natural language processing in the interactive drama facade. *Proceedings of the 1st International Conference on technologies for Interactive Digital Storytelling and Entertainment*, 2003.

[9] M. J. Nelson and M. Mateas. Search-based drama management in the interactive fiction Anchorhead. *Proceedings of the First Artificial Intelligence and Interactive Digital Entertainment Conference*, 2005.

[10] F. Peinado and P. Gervas. Transferring game mastering laws to interactive digital storytelling. *Proceedings of the 2nd International Conference on Technologies for Interactive Digital Storytelling and Entertainment*, 2004.

[11] J. Porteous and M. Cavazza. Controlling narrative generation with planning trajectories: the role of constraints. *Proceedings of 2nd International Conference on Interactive Digital Storytelling*, 2009.

[12] M. Riedl and R. M. Young. From linear story generation to branching story graphs. *IEEE Journal of Computer Graphics and Animation*, 26(3):23–31, 2006.

[13] M. O. Riedl, A. Stern, D. M. Dini, and J. M. Alderman. Dynamic experience management in virtual worlds for entertainment, education, and training. *International Transactions on Systems Science and Applications*, 2008.

[14] D. L. Roberts, M. J. Nelson, C. L. Isbell, M. Mateas, and M. L. Littman. Targeting specific distributions of trajectories in MDPs. *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, 2006.

[15] A. Rollings and E. Adams. Andrew rollings and ernest adams on game design. *New Riders Press*, 2003.

[16] M. Sharma, M. Mehta, S. Ontanon, and A. Ram. Player modeling evaluation for interactive fiction. *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment conference*, 2007.

[17] X. Su and T. M. Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in Artificial Intelligence*, 2009.

[18] D. Thue, V. Bulitko, M. Spetch, and E. Wasylishen. Interactive storytelling: A player modelling approach. *Proceedings of the third Artificial Intelligence and Interactive Digital Entertainment Conference*, 2007.

[19] M. E. Tipping and C. M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society*, B61(3):611–622, 1999.

[20] P. W. Weyhrauch. Guiding interactive drama. *Ph.D. Dissertation*, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA. Technical Report CMU-CS-97-109, 1997.

[21] S. Zhang, W. Wang, J. Ford, and F. Makedon. Learning from incomplete ratings using non-negative matrix factorization. *Proceedings of the 6th SIAM Conference on Data Mining*, 2006.

Evaluating the Models and Behaviour of 3D Intelligent Virtual Animals in a Predator-Prey Relationship

Deborah Richards[1], Michael J Jacobson[2],John Porte[1], Charlotte Taylor[2], Meredith Taylor[1], Anne Newstead[2], Iwan Kelaiah [1], Nader Hanna[1]

[1]Department of Computing
Macquarie University
North Ryde, NSW, 2109, Australia
+61 2 9850 9567
deborah.richards@mq.edu.au

Centre for Computer Supported Learning and Cognition, Faculty of Education and Social Work,
The University of Sydney, NSW, 2106, Australia
+61 2 9036 7671
michael.jacobson@sydney.edu.au

ABSTRACT

This paper presents the intelligent virtual animals that inhabit Omosa, a virtual learning environment to help secondary school students learn how to conduct scientific inquiry and gain concepts from biology. Omosa supports multiple agents, including animals, plants, and human hunters, which live in groups of varying sizes and in a predator-prey relationship with other agent types (species). In this paper we present our generic agent architecture and the algorithms that drive all animals. We concentrate on two of our animals to present how different parameter values affect their movements and inter/intra-group interactions. Two evaluations studies are included: one to demonstrate the effect of different components of our architecture; another to provide domain expert validation of the animal behavior.

Categories and Subject Descriptors

I.2 ARTIFICIAL INTELLIGENCE;I.6 SIMULATION AND MODELING, I.6.3[**Applications**] I.6.7 [**Simulation Support Systems**] *Environments*

General Terms

Algorithms, Measurement, Design, Experimentation.

Keywords

Agents, artificial life, boids, educational virtual worlds, biology education, science inquiry.

1. INTRODUCTION

Understanding the nature of and processes involved in scientific inquiry is an important skill that is difficult for most school students to acquire. This is an important challenge as inquiry figures pivotally in many national science plans. Key inquiry skills include a wide range of activities involved in scientific research such as hypothesis formation, experimental design, data collection and analysis, evaluation and reflection on the quality of evidence for hypotheses. Yet despite the aspirations for curriculum reform expressed in science policy documents, the practice of science education does not generally provide significant opportunity for students, especially at primary/middle school and secondary school, to experience genuine scientific inquiry [1].

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

The goal of our overall program of research is to develop innovative learning technologies that consist of 3D virtual worlds with embedded agent architectures. These educational "VWorlds" provide "virtually" authentic contexts for students to engage in scientific inquiry practices as they learn about biological systems, such as problem-identification, making observations and drawing inferences, interviewing characters, and collecting and analyzing data. Our project involves multidisciplinary collaboration with researchers in computer science and graphics, learning and cognitive sciences, and biology, as well as classroom science teachers. In this paper, we focus on the agent architecture we are developing by describing the artificial animals that inhabit our VWorld. We begin with consideration of related research.

2. RELATED RESEARCH

There is considerable interest in the use of computational modeling in modern biological research [15]. In particular, agent-based modeling (ABM) techniques (sometimes referred to as "individual-based models" in the literature) have been used to model a variety types of biological phenomena, such as flocking behaviors of birds and fish [8], synchronous firefly flashing [14], and the dynamics of predator-prey interactions in ecosystems [6]. Topping et al. [13] use ABMs to model an entire ecosystems in the animal, landscape and man simulation system (ALMaSS) that allows policy decisions to be made and includes many vegetated and non-vegetated areas, a range of crops with multiple growth models and multiple animals. Interactions between species are minimal in their ABM. Siebert, Ciarletta and Chevrier [10] are also interested in modeling complex systems. However, rather than creating a multi-agent system (MAS), they simulate a co-evolution where each agent type (sheep, grass and wolves) is a separate model/system connected via a coupling artifact.

In the intelligent virtual agent/virtual world research space, foundational work was done by [3]. More recent and specific to intelligent animals concerns deer with an artificial nose that detects the emotions of other conspecifics [2] and gray wolves that begin as pups and overtime develop certain social behaviours through learning to express age appropriate emotional states involving context-specific emotional memories [12]. Unlike our virtual world, these studies only concerned one type of animal.

In terms of the overall learning technology environment we are developing, research exploring the nature of learning with multi-user virtual worlds and 3D game environments has documented interesting educationally relevant outcomes, such as their motivational power and the opportunity to help develop important skills (e.g., collaboration [11]). The learning design features of

our VWorld build upon earlier agent-augmented multi-user virtual environments research [5] that were found to deliver significant benefits in science secondary classrooms.

3. OUR VIRTUAL WORLD - OMOSA

Omosa is a fictitious world that allows students to gain science inquiry skills and explore scientific concepts about biological systems. We chose not to model a specific place, flora, or fauna as we did not want the concepts learned to be restricted to the context we provided. To create Omosa, we needed to balance the level of detail in our environment with the complexity of our animals and human agents in a virtual environment with real-time graphics. We are using multi-platform game development software called Unity3D (http://unity3d.com/), which contains in-built features to reduce complexity while maintaining appearance such as lightmapping and occlusion culling.

We put several locations on Omosa (see Figure 1) where students can collect information and complete learning activities. These areas are: the village (where the indigenous Omosans live); the hunting ground (where our animals are located); the research lab (where students can collect information on ecological research and speak to an ecologist); and the weather lab (where students can collect information on climate research and speak to the climatologist). Artefacts can be collected in each location.

Figure 1. Our virtual world, Omosa

We modeled all our structures using Blender (http://www.blender.org/) to keep the polygon count as low as possible. We used Mixamo (http://www.mixamo.com/) to design and purchase low polygon human models. From TurboSquid (http://turbosquid.com/) we purchased three extinct animals (Andrewsarchus, Bluebuck, and Indricotherium) and an Iberian Lynx. In this paper we focus on two of our animals: the Bluebuck and Andrewsarchus, which we call a Yernt (one is laying on the ground in Figure 2) and a Tooru (three are feeding on a Yernt in Figure 2); the Yernt is the prey and the Tooru its predator.

Figure 2. Tooru (Predator) and Yernt (prey)

4. ARCHITECTURE

Our animals are agents who are embodied and situated in the Omosan environment. Each animal has its own state but shares its behaviour and population parameters with other animals of the same species (i.e. flockmates or conspecifics). As well as knowledge of its own state, each animal has access to lists of its predators, prey and flockmates. Each agent acts autonomously seeking to satisfy the goals determined by a combination of parameters introduced in this section.

In this section we present our model parameters and agent states and describe how the agents reason to decide what action to perform (e.g. chase, flee, eat) and the direction to move in.

4.1 Flocking - Tweaking the Boids algorithm

Reynolds [8] suggested that the seemingly complex group behavior seen in flocking can be modeled when individuals (boids) are driven using a small number of simple rules. A basic Boid algorithm includes: separation (or collision avoidance with nearby flockmates), alignment (or velocity matching with nearby flockmates), and cohesion (or centring by staying close to nearby flockmates).

The SeparationVector is a direction vector that is calculated and achieved at the individual boid level. If any other boid is too close then the SeparationVector will steer the boid away. Given a desired spacing, the distance to all other boids is measured. If *distance* < *spacing* then a vector can be calculated such that *boid1_position* – *boid2_position* = *SeparationVector*. This SeparationVector is a *xyz* direction that the current boid now intends to travel in order to maintain a distance from other boids. If multiple boids fall within the desired spacing then the resulting SeparationVectors can be summed together.

The AlignmentVector is a direction vector that is calculated at the entire boids group level. It is the average direction that the entire group of boids is travelling in.

The CohesionVector is a direction vector that is calculated at an individual level. It points from the current boid towards the average position of all other boids.

These three vectors can be summed to produce an output vector, the direction the boid will finally move, that represents the intentions of the boid. To represent the unpredictability of individuals (1) includes a RandomVector, as follows:

OutputVector = *SeparationVector* + *AlignmentVector* + *CohesionVector* + *RandomVector* (1)

This random value could be replaced by a probability if a suitable stochastic model was identified for that animal type (i.e. species, gender, age, etc). Also, greater or lesser importance can be applied to any of the input vectors by multiplying them by a weight. For example in (2) we increase the importance of grouping together with:

*OutputVector = SeparationVector + AlignmentVector + (CohesionVector * 1.5) + RandomVector* (2)

The individual boid will now move in the OutputVector direction from its current location. To avoid collisions (3) builds upon this algorithm as follows:

OutputVector = SeparationVector + AlignmentVector + CohesionVector + RandomVector + ObstacleVector (3)

Where ObstacleVector points away from a tree or a rock that an individual boid is getting too close to and would prefer to not crash into.

Finally, in Omosa we do not want the entire population for each animal to behave as a single group. For each type of agent our model allows us to specify the size of subgroups within a population. In our implementation, herding is achieved by modifying both the AlignmentVector + CohesionVector to only consider the nearest HerdSize boids. In this way we have sub groups of boids that will dynamically readjust itself to only use the nearest boids. Different size herds can be seen in Figure 3.

Figure 3. Yernt and Tooru Boids

4.2 Beyond Boids – Predator/Prey agents

The animals in Omosa, as in real ecosystems, do more than move around; they exhibit behaviours such as growing, dying, hunting, eating, etc. Here we focus on the predator-prey relationship which drives many of the group and individual behaviours. To achieve this we have developed a predator model and a prey model. Figure 4 depicts how these models influence the boids.

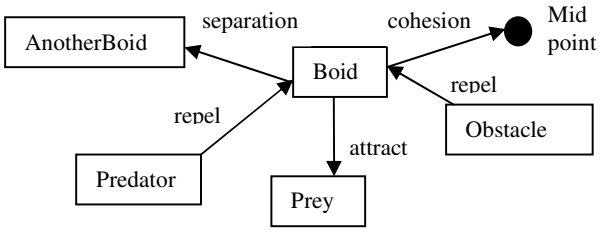


Figure 4. Simple model of factors influencing the individual boid.

Prey Model –This model produces a vector (4) calculated at an individual level that points towards prey animals. In order to indicate urgency some prey animals will be given greater importance, or weight, for a number of reasons:

- The closer the prey the greater the weight. This simulates the predator singling out a target as it bears down on it.
- The more fatigued the prey the greater its weight. The predator attacks the weak.
- The more injured the prey the greater the weight. The predator attacks the weak.

PreyDistance, PreyFatigue, and *PreyHealth* are all values between 0.0 and 1.0.

*PreyVector = PreyDirection * (1.0 – PreyDistance) * PreyFatigue * (1.0 – PreyHealth)* (4)

Predator Model – In contrast to the Prey model, this model produces a vector (5) calculated at an individual level that points away from predator animals. In order to indicate urgency some predators are given greater importance for the following reasons:

- The closer the predator the greater the weight. This simulates the prey fleeing for its life.
- The more threatening the prey the greater the weight. This simulates some animals or even human hunters being more dangerous than others, and the prey reacting accordingly.

PredatorDistance and *PredatorThreat* are values between 0 & 1.

*PredatorVector = PredatorDirection * (1.0 – PredatorDistance) * PredatorThreat* (5)

Figure 5 shows how each of the components in our architecture fit together. We can see a pipes and filters like structure between the Flocking, Predator and Prey components which allow the agent to achieve its decision making goal about which direction to go in. This decision is influenced by FollowVector, a vector that behaves much like a leash. Depending on whether they are hunting or resting we can adjust FollowVector to migrate the entire group from one location on the map to another. We can also adjust the weight of this leash to ensure the group does not go running off into the ocean or another area we want them to avoid. In the future, this vector could be replaced with a subsystem that intelligently determines locations of herds both for initial spawning and migration purposes.

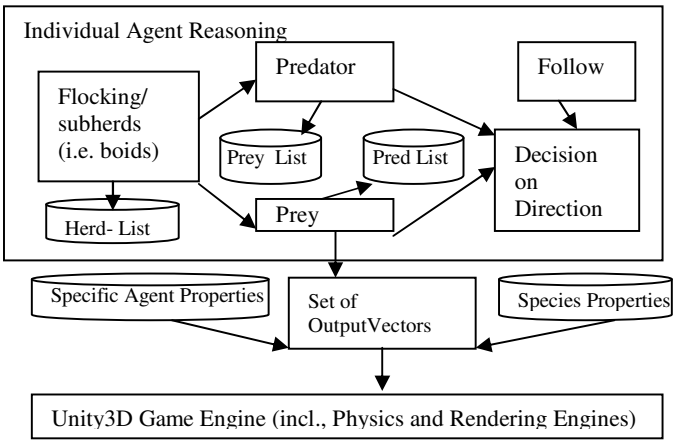


Figure 5. Omosa Architecture and Agent Reasoning.

The Individual Agent Reasoning System is used to determine the direction that the individual agent intends to go. However, our agents have additional restrictions that will define the eventual *OutputPosition*. These other factors are seen as being beyond the control of the agent and not part of their reasoning. For example, the physics engine will slow agents down as they attempt to ascend hills and accelerate agents as they descend.

Another factor applied after the agent has made their own decision regarding their intended direction is maximum speed, a value defined by the type of animal represented, which is further limited by the agent's health, fatigue, stamina and age. This models the fact that animals, including humans, are externally limited by these factors, in addition to their internal influence on the individual decision making as described in the prey model presented. To provide a more natural model of klinokenesis (change in direction) in our animals we have developed a smoothing algorithm that allows the animal to adjust speed and angle to provide a more rounded trajectory rather than a 180 degree turnaround which can result in strange behaviours in conjunction with the physics engine. For the same reason, we also adjust the animals speed to slow down when approaching another agent/object. A summary of individual and group parameters, states and behaviours is given in Table 1.

Table 1. Agent parameters, states and behaviours

Intra-agent (conspecific) parameters	**Inter-agent (different species) parameters**
Cohesion – individual	Separation – individual
Alignment – group	Obstacle - individual
Follow – group	Prey – individual
Obstacle - individual	Predator - individual
Individual States	**Group/Species States**
Health, Stamina, Life Stage (i.e. birth, mature, dying which affects size & colour), Urgency, Threat level, Location	Population/Herd size, Life Expectancy, Stage duration, Spacing, Perceptual Distance, Speed, Health Regeneration
Individual Behaviours	**Group Behaviours**
Roaming, Hunting, Standing, Feeding, Fleeing, Dying, Birth	Hunting/Stalking

As a group the animals work together to hunt down prey and avoid obstacles and predators, while as individual agents they maintain their own goals and states. While some of the group behaviours are very efficient, as the individual characteristics became more complex and specific, it has been necessary to find ways to maintain performance. For example, not all behaviours need to be refreshed every cycle to create believability. Performance is discussed again in the conclusion section.

5. EVALUATION STUDIES

At stated in our introduction, the goal of our project is to provide experience in conducting scientific inquiry and improve knowledge of biological concepts in secondary schools. In this section we present condensed results from two evaluation studies. The first study seeks to verify our models, algorithms and architecture as presented in the previous section. The second study seeks to validate our approach through an interview with an expert ethologist who has not been involved in the project.

5.1 Study 1 – Model Verification

In the first study we have collected data which evaluates the components in our Individual Agent Reasoning System. The design is presented next, followed by results and discussion.

5.1.1 Design

To evaluate the effect of the flocking, predator and prey components on the behavior of our animal, we have collected data about our predator and prey populations over a 20 minute period using different combinations of components in our architecture. Each run/simulation used identical population parameter settings. The parameters used were the default settings for each population identified by the biologist on our team as most appropriate for our predator and prey population. For each run, we collected the total population, number of births and deaths for both prey and predators as well as the number of predator kills and prey deaths from old age. The six runs reported in this paper include:

1. Default/Complete: Flocking/herding, Predator/Prey awareness.
2. No flocking/herding (1 minus Boids model)
3. No predator awareness (1 minus Predator model)
4. No prey awareness (1 minus Prey model)
5. No prey or predator awareness (1 minus Prey and Predator)
6. No subherds (1 minus herds, i.e. influenced by entire flock not just neighbours/herd members).

5.1.2 Results and Discussion

The results of data analysis are shown in Figures 6-11. The comments below are based on review of those figures as well as observations of agent behavior on the screen during each run. We are particularly interested in the kill rates, as the domain expert (see next subsection) equated success with natural/low kill rates.

The data in Figure 6 is based on the complete model presented in the previous section and includes flocking, predator / prey awareness and herding. We observe normal agent behavior and a fairly balanced system. The birth rate maintains the prey population. The predator made 13 kills over 20 minutes.

In Figure 7 the flocking component was turned off, although obstacle avoidance is included to avoid collisions with each other. Without any flocking enabled the agents still functioned surprisingly well. Prey were able to escape predators on most occasions. The level of realism seemed to be reduced, the prey behaved somewhat like water trickling between the predators, but definitely not as a group. Prey population fluctuated but was near maximum after 20 minutes. Predator made 12 kills in 20 minutes.

In Figure 8, when there is no predator awareness (i.e. prey does not respond to predator), we see that the prey were wiped out in less than 2 minutes. Prey did not attempt to evade the predator. We observed that the predator had some difficulty getting to all of the prey because there were too many prey carcasses in the way. Prey were unable to reproduce and maintain population. Predators made 80 kills over 2 minutes. Note that this simulation is not realistic: there would be an upper limit on how many prey a predator seek to kill; the predator population will die out when the food source is gone.

In Figure 9 there is no prey awareness (i.e. predators do not respond to prey); the predator completely fails to function. The prey was aware of the predators when they moved to the hunting area, but just moved to a safe distance. Prey population fluctuates due to life span, birth rate is able to maintain maximum population. Predator made 0 kills in 20 minutes.

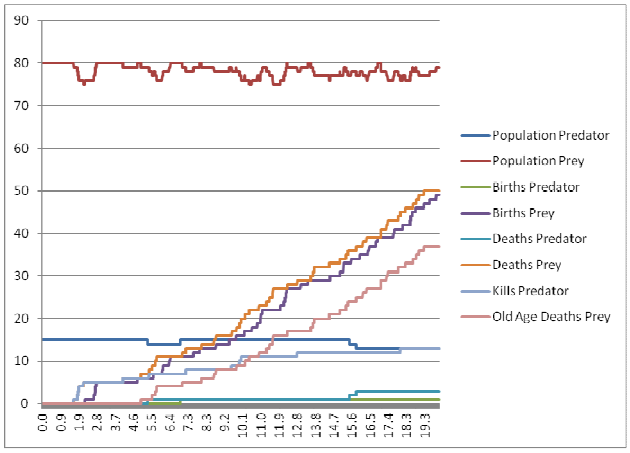


Figure 6. Default Settings (Flocking, Predator / Prey Awareness, Herding) Normal behaviour.

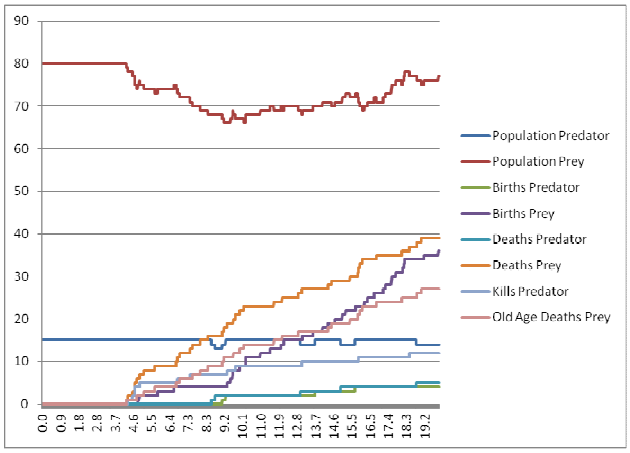


Figure 7. No Flocking (Boids still avoid collisions with each other).

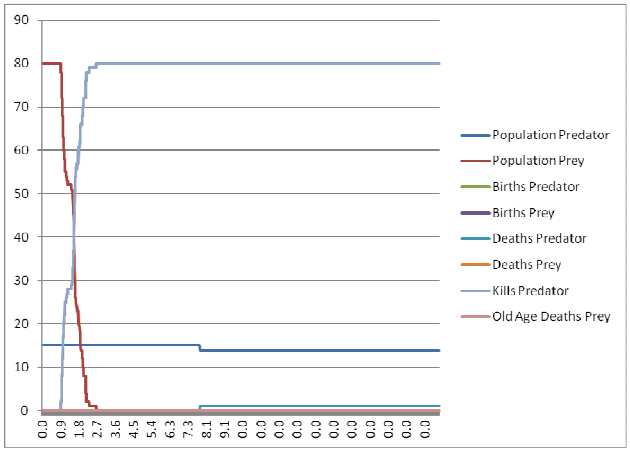


Figure 8. No Predator Awareness (Prey does not respond to predator)

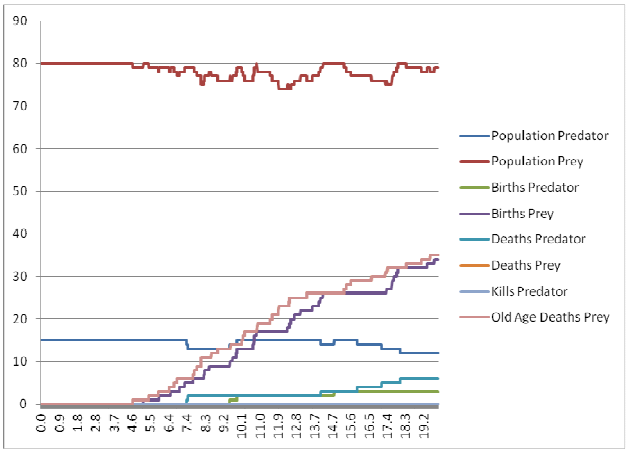


Figure 9. No Prey Awareness (Predator does not respond to prey)

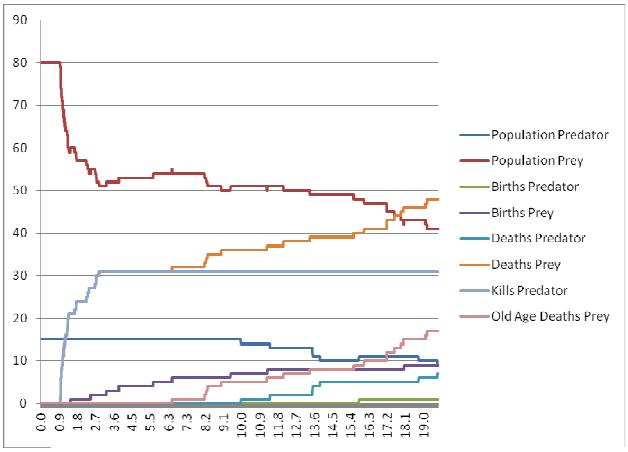


Figure 10. No Predator Vector and No Prey Vector (Both predator and prey don't respond to other).

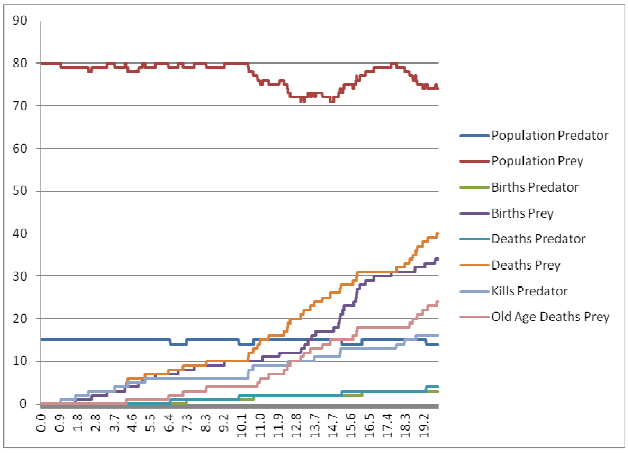


Figure 11. No Sub Herds (Flocking is enabled, boids are influenced by all of their own type, not just those nearby)

In Figure 10 the predator and prey models are both switched off. (Both predator and prey don't respond to other). Predators perform badly as in the previous condition. However at approximately one minute the predator herd happens to walk right into the prey herd, which in turn doesn't respond and takes a lot of losses. After that the predator never came close to the prey again, instead spending most of the time splashing around on the beach. Predator made 31 kills over 20 minutes.

Finally in Figure 11 we evaluate turning off the herding feature so that no subherds exist within a population. Flocking is still active but boids are influenced by all of their own type, not just those nearby). Both predator and prey function well. Predators seemed to benefit from this setting, showing more cohesion, while the prey may have been hampered from too much influence from others of their species (instead of just relevant ones nearby). Prey population fluctuated but was maintained. Predators made 16 kill.

We conclude that the predator and prey models are essential to model natural success/kill rates. Though success rates are mostly unaffected by flocking/herding they are necessary to provide a realistic 3D simulation of animals which live and act in groups.

5.2 Expert Validation

In addition to having access to a number of advisers, one of the investigators in this research project is a biologist. However, to provide independent evaluation of our animal behaviours we approached an expert in the field of animal communication and conservation, whose particular area of expertise and interest was ungulates, with a focus on elk and bison. We conducted a one (1) hour interview involving demonstration of our system and a series of structured questions. The steps we followed and questions posed, together with responses are described below.

5.2.1 Step 1: First Impressions

In order to put our animals in their current environmental context and to gain first general impressions, we began with a tour of the village including a conversation with an Omosan hunter and then we used the game/site map (Figure 1) to locate animals that we then introduced up close, as in Figure 2. We then took a bird's eye view of a part of Omosa containing herds of both animal types. Our first question was simply "What are your first impressions?"

We were unaware of the expectations of our expert, and she was unaware of what she would see. Thus, as a first response she commented that [real] animals are predictable but ours are unpredictable. She mentioned the *many eyes hypothesis* where the animals would stay in groups, always with some animals watching while others fed/grazed. The expert's response regarding unpredictability occurred after a relatively lengthy initial period where the two populations had been grazing independently and then appeared to become aware of one another. This time delay can be seen in Figure 12. Figure 12 shows a plot of the agent movements over time. The red (left side) path shows predators. The green (right side) show prey. We can see over time that the separate populations begin to interact and that the two populations are moving regularly with predators following prey and prey tending to flee from predators as shown more clearly in the pathways at the bottom of the figure.

As the behaviours became more interesting, including predators circling a wounded or dead prey, which our expert said is like wolf pack behavior, the expert became more engaged and excited, sometimes intrigued by the behaviours observed. When students use the world, the animals will have potentially been through numerous life cycles and it is unlikely they will be at the initial and less interesting reset/spawning stage.

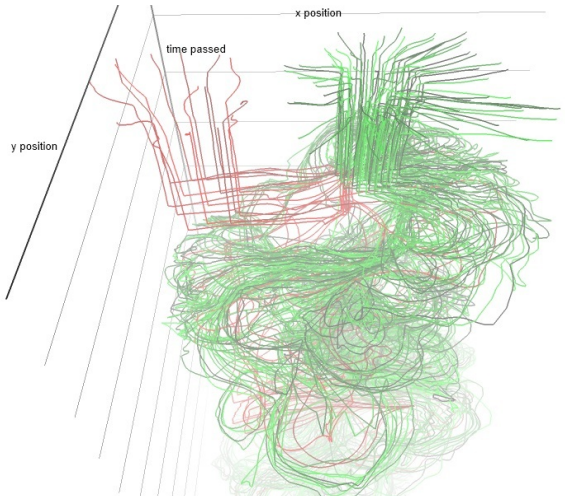


Figure 12. 3D Model of animal paths showing agent position over time(red/left=predator, green/right=prey

5.2.2 Step 2: Parameter Testing

To allow model adjustment and assist learning, we allow the group level parameters (see Table 1) to be adjusted via sliders. We asked the expert to select up to 3 parameters that they would like to change, though only one a time. We asked her to make a prediction for each parameter before any changes were made. Our expert was only interested in changing two parameters: speed and perceptual distance and stated that prey can move faster and have further perceptual distance than predators. The other parameters, such as stamina, were perceived to be secondary. The expert stated that there should be at least two or three times as many prey than predators and was happy with our relative herd sizes of 3 for predators and 15 for prey. In terms of population sizes of 80 for prey and 15 for predator, the biologist was also satisfied.

Speed was changed first by increasing the speed of the prey and reducing the speed of the predator. The prediction was there would be less kills/success and the animal behaviours would be more lifelike. In accordance with the prediction, there were considerably less kills. Surprisingly we also observed that the predators seemed to be moving as one towards the prey rather than in herd sizes of 3 towards selected prey and this inefficient behavior would have affected success rates. The expert added that a kill success rate of 10% was normal for natural populations but that speed should be slightly unnatural resulting in greater numbers of kills so that the simulation is not too boring.

The second parameter to be changed was perceptual distance. In line with natural differences, the predator value was decreased to 20 and prey value was increased to 35. Again the prediction was that the behavior would be more natural and would result in less kills/predator success. Indeed less kills were observed. What was not predicted and was quite surprising was the opposite behavior to our change relating to speed. Even though we left the speed settings to those specified by our expert, this time rather than predators appearing to act as a whole population moving slowly

towards the prey, the majority of Tooru continued to ignore the Yernt. Only individuals at the edge of the predator group closest to the prey group appeared to notice the prey and run off in that direction, leaving their herd (the other two) behind. It appears that the individual had come within the perceptual distance allowing them to recognize the prey, and was quite hungry by that time pulling them more strongly towards the prey than their mates. The mates who had not been able themselves to see the prey, were still close enough to other conspecific herds and thus they joined the new herd to satisfy that need. The increased perceptual vision of the prey in detecting the predator resulted in the flock of prey moving away from the predator herds/population, making it increasingly difficult for the predator to spot them.

5.2.3 Step 3: Rating of Environment

We chose to use the questions from [2] to "establish the contribution of behaviours to the perceived realism of the animals within the environments and the contribution to the overall experience" (p.155). However, we sought to validate our complete architecture, system and the emergent behaviors with a domain expert rather than test alternative models/combinations of system components to subjects (e.g. no-flocking, no flight, no fear/emotion, etc) with immersive technology students. Thus we did not use Likert scale responses but allowed the expert to use their own term. Table 2 shows the questions and brief answers.

Table 2. Parameters defining our three current conspecifics

Question	Response
1. How realistic was the graphical representation of the animals in the environment?	Good
2. How much did the environment engage you generally?	Very
3. How much did the animals add to the realism of the environment?	Very
4. Did the animals seem alive in the environment?	Yes
5. Did the animals appear to be behaving in an intelligent manner?	Depends
6. How realistic was the behaviour of the animals?	Good
7. How quickly did you adjust to the virtual environment experience?	Immediate
8. To what extent did the animals seem to be reacting to their environment?	Good
9. To what extent did the animals appear to be reacting with one another?	Good
10. To what degree did the animals appear to make an emotional reaction?	Motivation observed

Regarding whether the animals were perceived as alive, (Q4) the expert added that "movement is critical, it brings the animal to life and the animals bring the virtual world to life". Regarding whether the animals appeared to be behaving in an intelligent manner (Q5) the answer was qualified by saying that it depends on what parameters are used. The expert was "bothered by lack of group cohesion within the pack" which was not always evident. However, the circling of a pack around prey and chasing after the same prey could be observed at times. We asked the question "How compelling was your sense of moving around inside the virtual environment?" but it was not answered due to lack of relevance as the expert did not control the initial tour of the world and watching the behaviours did not involve interaction with the animals.

Regarding questions 8 and 9, the expert commented that the flocking indicated awareness of conspecifics; prey were aware of predators and reacted by fleeing when they were chased. Chasing was evidence of predators being aware of prey and reacting to them. The fact that at times both animal agent types grazed and showed no interest in the other animal type indicated that there were also other factors affecting their interest in and desire to hunt or flee. However, for some settings a lack of cohesion on the part of predator was observed. At these times it seemed that predators were not reacting to one another even though prey did react to one another.

Though our model does not explicitly include emotion (fleeing could be triggered by fear), we included the question from [20] regarding emotion to test and provide opportunity for discussion whether the ethologist had endowed our animals with emotional behavior or believed that emotional factors should be modeled. The expert stated outright that they were uncomfortable with the word "emotion" when considering the behavior of animals. They preferred the term and concept of motivation. The expert observed that the prey were motivated to avoid the predators, which could be seen as due to fear, but they did not feel it necessary or appropriate to attribute emotion as the cause.

5.2.4 Step 4: Usefulness for Education

The goal of our intelligent animals is to allow students to observe animals in a natural setting to see how they may behave, allow them to set various hypotheses about the animals and phenomena occurring in Omosa and to teach them about complex systems. It was not our goal to provide ecologically sound and complete animal models which would allow us or others to make decisions and predications about these populations in the real world. Thus, we asked "Do you believe the world would be useful for educational purposes?' They responded "Definitely, it would get the students engaged".

When asked if the world would be useful at the tertiary level, perhaps in some of their own teaching context, they were more hesitant and remarked with respect to the animals that it could be useful if more parameters could be made available (though none were specifically suggested) and students would need to be able to change them. The expert suggested that Omosa 2.0 would be needed for tertiary biology students. When asked what would be in 2.0, they suggested multiple prey types and predator switching between prey depending on factors such as availability.

5.2.5 Step 5: Additional Features and Directions

Throughout the interview a number of behaviours were suggested for possible inclusion, as follows:

- Reproduction rates influenced by success rates,
- Targeted kills, e.g instead of attacking many/closest prey, predators would intelligently pick one or two, e.g. smallest. (we already factor in health).
- Complementary/coordinated group behaviours, e.g. some prey-flockmates would come back and defend, some pack members may not join in.
- After killing predators go back to foraging (which we do).
- Might need to change life span.

These features and others are considered as further extensions to our agents in the next section.

6. CONCLUSIONS & FUTURE WORK

Animals have provided agent researchers with so called biologically-inspired solutions to issues such as coalition formation (e.g. [4]) and other social dilemmas involving communication, coordination and cooperation to solve problems such as load balancing, message congestion and bandwidth allocation. Similarly, we anticipate that software/network agent research related to the handling of social interactions, decision making, self-interested agents and cooperation (e.g. [9]) could potentially offer some insights and extensions to our animal agents. As demonstrated, behaviours which simulate group communication and coordination exist in our model, however, to produce more lifelike animals we may want to extend our models with natural communication methods involving gestures and sound, similar to the use of the scents and an artificial nose [2]. MAS-based group decision-making may be a feature that our animal or human agents will need as in the study by [16]. Inclusion of updating schemes which allow the evolution of our models is also potentially attractive.

Currently, we simulate different life stages through size of the animal and intend to use changes in colour as a feature to indicate age. Also, while we have different values for traits for different species we do not currently have separate traits for different species or differentiate between behavior in males and females as in ALMaSS [13]. We would like to include stochastic elements into our models to potentially provide more authentic behaviours in determining initial locations to spawn animals and affecting whether a kill is successful or not. At this stage, we do not believe that explicit modeling of emotions is appropriate or necessary for our animals. We will conduct further studies using our models and more of our animals, as suggested by the expert, involving multiple predators and alternative prey.

Scalability is an issue facing both graphics researchers and agent researchers involved in building complex cognitive architectures and multi-agent platforms [7]. On the graphics side we have paid close attention to polygon counts. For example, using MeshLab (http://meshlab.sourceforge.net/) we were able to reduce the number of polygons in purchased animal models from around 6000 to no more than 1800 each. To support both the processing requirements of our agent reasoning approach with the processing requirements of the graphics, we have increased the number of frames between each animal in the herd updating its behavior. While this slightly decreases the realism of the animals' behavior, it significantly improves the overall game performance. As the complexity of the models and agents in Omosa increase, we will have to consider more strategies for maintaining the balance between processing speed and environment complexity.

Initial trials with teachers and a Science special interest group were enthusiastically received and led to modifications to Omosa involving the dialogue engine, interaction controls and smoothing of animal movement transitions. In November 2011 we began testing our workbooks and lessons in the classroom over 4 lessons. We are currently processing student data including measurements of learning gains and changes in levels of interest in science inquiry. Results will appear in a future publication.

7. ACKNOWLEDGMENTS

This project is funded via ARC Discovery DP1093170

8. REFERENCES

[1] Campbell, T., Abd-Hamid, N., Chapman, H. (2010). Development of Instruments to Assess Teacher and Student Perceptions of Inquiry Experiences in Science Classrooms. Journal of Science Teacher Education 21, 13–30.

[2] Delgado-Mata, C., Martinez, I J., Bee, S., Ruiz-Rodarte, R. and Aylett, R. 2007, On the Use of Virtual Animals with Artificial Fear in Virtual Environments New Generation Computing, 25(2): 145-169.

[3] Funge, J., X. Tu, & D. Terzopoulos 1999. Cognitive Modeling: Knowledge, reasoning and planning for intelligent characters,", Proc. ACM SIGGRAPH 99 Conference, Los Angeles, CA, August, 1999, 29–38.

[4] Haque, M., Rahmani, A. and Egerstedt, M. 2010. Biologically inspired coalition formation of multi-agent systems. In Proc. AAMAS '10, Vol. 1, 1427-1428.

[5] Jacobson, M. J., Miao, C., Kim, B., Shen, Z., & Chavez, M. 2008. Research into learning in an intelligent agent augmented multi-user virtual environment. In Proc. Int.l Conf. on Web Intelligence & IAT, 348–351.

[6] Levin,S.A., B.T.Grenfell, A.Hastings, & A.S.Perelson. 1997. Mathematical & computational challenges in population biology & ecosystems science. Science, 275: 334-343.

[7] Navarro, L, Flacher, F. and Corruble, V. 2011, Dynamic Level of Detail for Large Scale Agent-Based Urban Simulations, In Proc. AAMAS'2011, 701 - 708.

[8] Reynolds, C. 1987. Flocks, herds and schools: A distributed behavioral model.SIGGRAPH'87,ACM,Anaheim,CA,25–34.

[9] Salazar, Rodriguez-Aguilar, Arcos, Peleteiro,Burguillo-Rial, Emerging Cooperation on Complex Networks, In Proc. AAMAS'2011, 669-676.

[10] Siebert, J., Ciarletta, L., and Chevrier, V. 2010. Agents and artefacts for multiple models co-evolution: building complex system simulation as a set of interacting models. In Proc. AAMAS '10, Vol. 1, 509-516.

[11] Steinkuehler, C. A. (2004). Learning in massively multiplayer online games. In Y. B. Kafai, W. A. Sandoval, N. Enyedy, A. S. Nixon, & F. Herrera (Eds.), Proc. 6th Int. Conf. Learning Sciences Mahwah, NJ: Erlbaum, 521-528.

[12] Tomlinson, B and Blumberg. B2002. Synthetic Social Relationships in Animated Virtual Characters." In: From Animals to Animats 7. Proc. 7th Int. Conf. on the Simulation of Adaptive Behavior (SAB '02). Edinburgh, UK.

[13] Topping, C.J., Hansen, T.S., Jensen, T.S., Jepsen, J.U., Nikolajsen, F. & Odderskær, P. 2003: ALMaSS, an agent-based model for animals in temperate European landscapes. - Ecological Modelling 167(1-2): 65-82.

[14] Wilensky, U. and K. Reisman, 2006. Thinking like a wolf, a sheep or a firefly: Learning biology through constructing and testing computational theories -- an embodied modeling approach. Cognition & Instruction. 24(2): p. 171-209.

[15] Wooley, J.C. and H.S. Lin, 2005. Catalyzing inquiry at the interface of computing and biology, Washington, D.C.:NAP.

[16] Zappala, J. 2008. Multi-agent simulation of group decision making in animals, MSc Thesis, Uni. of Nottingham.

Model of the Perception of Smiling Virtual Character

Magalie Ochs
CNRS-LTCI Télécom ParisTech
ochs@telecom-paristech.fr

Catherine Pelachaud
CNRS-LTCI Télécom ParisTech
pelachaud@telecom-paristech.fr

ABSTRACT

A smile may convey different communicative intentions depending on subtle characteristics of the facial expression. Moreover, during an interaction, the expression of smile impacts on the observer's perception of both the social stance of the speaker and of the content of the talk. In this paper, we describe a perceptual study where we explore the effects of virtual characters displaying different types of smiles (namely politeness and amusement) when speaking on the user's perception. Based on the collected data, a model to automatically compute the user's potential perception of the virtual character's social stance depending on its smiling behavior and on its gender has been proposed.

Categories and Subject Descriptors

H.5.1 [**Multimedia Information Systems**]

General Terms

Algorithms

Keywords

Virtual character, smile, user's perception

1. INTRODUCTION

During dialog, non-verbal behaviors play an important role on interlocutor's perception. The content of the message but also the global stance of the speaker may be perceived differently depending on her gestures, her posture, and her facial expressions. For instance, smiles may enhance the global perception of a person [6, 18, 27] and even of a virtual character [14]. In this paper, based on a human-centric approach, we propose to explore the effects of smiles on the perception that users have of a virtual character.

A smile is one of the simplest and most easily recognized facial expressions [9]. To create a smile, the two muscles zygomatic majors, on either side of the face, have to be activated. However, others muscles may be implied in an expression of smile. Moreover, a smile may have several meanings - such as amusement and politeness - depending on subtle differences in the characteristics of the smile itself and of other elements of the face that are displayed with the smile. These different types of smiles are often distinguishable during a social interaction. Recently researchers [26, 22] have shown that people are also able to distinguish different types of smiles when they are expressed by a virtual character.

A smiling virtual character improves human-machine interaction. For example it enhances the perception of the task to be done and how the character is perceived. It increases the motivation and enthusiasm of the user [14, 31]. However, an inappropriate smile (an inappropriate type of smile or a smile expressed in an inappropriate situation) may have negative effects on the social interaction [31].

In this paper, we present research that aims at identifying the effects of different virtual character's smiles on the user's perception of the virtual character. More precisely, we have investigated how polite and amused smiles displayed by speaking virtual characters alter the user's perception both of the content of a message and of the stance of the virtual character. We considered the types of displayed smile and the gender of the virtual character. For this purpose, we propose a *human-centric approach* to both identify the characteristics of smiles and their effects on perception. We have first identified the dynamic and morphological characteristics of different types of virtual character's smile. Our method was to collect a corpora of smiles directly created by users. Characteristic features of each smile types were extracted from the analysis of the corpora. An evaluation study has been conducted to validate the identified smiles in context. Secondly, we have developed a web application to collect the user's perception of virtual characters displaying different smiles when saying an utterance. Two types of smile have been considered: polite and amused smiles. Two virtual characters, a female and a male one, were used. The results have been used to propose a model to automatically compute how user's (potential) perception of the agent is influenced dynamically by the display of agent's smile. It is a first attempt toward being a Theory of Mind model. The Theory of Mind is the cognitive ability to understand others' actions and expressions within an intentional or goal-directed framework (i.e. the *intentional stance* [5]). In our work, based on a human-centric approach, we aim at modeling the user's Theory of Mind of the agent's social stances.

The paper structure is as follow. After giving an overview of existing works on humans' smiles and on virtual characters' smiles (Section 2), we introduce the method used to identify the different types of smile of virtual characters (Section 3). In Section 4, we present the web application

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

developed to collect the user's (potential) perception of a smiling virtual character. In Section 5, we present the resulting data and we introduce a model to compute the user's perception of a smiling virtual character during an interaction. We conclude and present perspectives of this research in Section 6.

2. RELATED WORK

In this section, we present existing research on the types and meaning of smiles (Section 2.1) and the effect on the observer (Section 2.1), both in human-human interaction (Section 2.1.1 and 2.2.1) and in human-machine interaction (Section 2.1.2 and 2.2.2).

2.1 Types and meaning of smiles

2.1.1 Human smiles

According to Poggi and Chirico [24], a smile may have two basic meanings: "a purely expressive meaning, an expression of pleasure, and a communicative meaning, the goal of showing friendly to other people". Smile can also replace a word: one can smile to say "hello" [24].

The most common type of smile is the *amused smile*, also called felt, Duchenne, enjoyment, or genuine smile. Another type, which is often thought of as the amused smile's opposite is the *polite smile*, also called non-Duchenne, false, social, masking, or controlled smile [11]. Perceptual studies [11] have shown that people unconsciously and consciously distinguish between an amused smile and a polite smile. Other smiles have been identified, as for instance embarrassed smiles. However, in the current paper, we focus on two smiles: the amused and polite smiles.

These different smiles are distinguishable by their distinct morphological and dynamic characteristics. Despite, no consensus exists on the morphological and dynamic characteristics of the amused and polite smiles, it is, in general believed that the orbicularis oculi (which refers to the Action Unit (AU) 6 in the Facial Action Coding System [10]) is more present in amused smile than in polite smile. The dynamic characteristics of the amused smile are the smoothness and regularity of the onset, apex, offset (describing the temporal course of the facial action) and of the overall zygomatic actions, the mouth opening. The duration of the smile lasts between 0.5 and 4 seconds [1, 9]. In the expression of a polite smile, the cheek raising (AU6) is absent, the amplitude of the zygomatic major (AU12) is small, the smile is slightly asymmetric, the apex is longer, the onset shorter, the offset is more abrupt than in amused smile, and the lips may be pressed [9].

2.1.2 Virtual smiles

In order to increase the repertoire of communicative behaviors of virtual character's facial expressions, several researchers have considered different virtual character's smiles. For instance, in Tanguy [30], two different types of smiles, amused and polite, are used by a virtual character. The amused smile is used to reflect an emotional state of happiness whereas a polite smile, called fake smile in Tanguy (2006), is used by the virtual character masking sadness with a smile. The amused smile is represented by lip corners raised, lower eyelids raised, and an open mouth. The polite smile is represented by an asymmetric raising of the lip corners and an expression of sadness in the upper part of the face. In Rehm and André [26], virtual characters mask a felt negative emotion of disgust, anger, fear, or sadness with a smile. Two types of facial expression were created according to Ekman's description [8]. The first expression corresponds to a felt emotion of happiness (including an amused smile). The second one corresponds to the other expression (e.g. disgust) masked by unfelt happiness. In particular, the expression of unfelt happiness lacks the AU6 activity and is asymmetric. It may correspond to a polite smile. Niewiadomski and Pelachaud [21] proposed an algorithm to generate complex facial expressions, such as masked or fake expressions. An expression is a composition of eight facial areas, each of which can display signs of emotion. For complex facial expressions, different emotions can be expressed on different areas of the face. In particular, it is possible to generate different expressions of joy: a felt and a fake one. The *felt* expression of joy uses the reliable features (AU6), while the second one is asymmetric.

Several other virtual characters smile during an interaction to either express a positive emotion [25], to create a global friendly atmosphere [31], or for salutation [4]. Generally, these virtual characters use only the amused type of smiles. In this present work, we explore different types of smiles a virtual character may perform.

2.2 Perception of smiles

2.2.1 Human-human interaction

Several studies have shown that individuals who smile are perceived more positively than non-smiling persons. Smiling people are viewed as more relaxed, kind, warm, attractive, successful, sociable, polite, happy, honest with a higher sense of humor, and less dominant [6, 7, 18, 20, 27].

In Western society, the women smile more than men and are also expected to do so [6, 17]. For instance, in Deutsch, LeBaron, and Fryer [6], a study of the perception of photography of male and female smiling and non-smiling faces show significant differences depending on gender. Whereas there is no significant difference between smiling men and women, the absence of smile for a woman seems to deteriorate her image compared to a man. Indeed, the study has shown that women who do not smile are perceived less happy and relaxed than non-smiling men. The hypothesis is that different standards are applied to evaluate non-verbal behavior of men and women. People expect that women smile more than men, and consequently, a deviation from that expected behavior influences negatively the perception of non-smiling women. No distinction between polite and amused smiles is considered in the study. Moreover, as shown in Hess, Blairy, and Kleck [12], since smile is expected for a woman, perceiver may not consider women's smiles as informative compared to men. Moreover, research has shown an influence of gender on the perception of the intensity of a smile: men's amused smiles are perceived as more intense than those of women.

Concerning the detection of different smile types, research has shown that women are more sensitive to non-verbal signs and more able to decode facial expressions cues, even for virtual characters' faces [15]. Women make more extreme judgment ratings than men when decoding facial expressions [13]. The type of displayed smile affects also the perception of the observer. For instance, people showing amused smile are perceived more expressive, natural, outgoing, sociable,

relaxed, likable and pleasant than when they show polite smiles [11, 17]. Amused smiling faces are also perceived as being more sociable and generous than polite smiling face [19].

2.2.2 Human-machine interaction

Several researchers have explored the effect of smiling virtual characters on the user's perception both of the character's social stance and of the speech content.

Effects of smiles on social stance.

In Krumhuber, Manstead, and Kappas [15], the results show that virtual characters displaying a felt smile (longer onset and offset) were rated as more attractive, more trustworthy, and less dominant than those showing a faked smile (a short onset duration). In Rehm and André [26], a perceptive test has enabled the authors to measure the impact of fake expressions of smile on the user's subjective impression of the character. The participants were able to perceive the difference, but they were unable to explain their judgment. The character expressing an amused smile was perceived as being more reliable, trustable, convincing, credible, and more certain about what it said compared to the character expressing a negative emotion masked by a smile.

In Krumhuber, Manstead, and Kappas [15], a gender effect has been noticed: smiles shown by female virtual characters are judged less authentic that those displayed by men, whatever is the smile.

Effects of smiles on speech content.

In Krumhuber, Manstead, Cosker, Marshall, and Rosin[14], the authors have explored the impact of different types of smile displayed by virtual faces on the users' perception of the virtual character's speech content. The context of the interaction is a job interview. The results show that the type of smiles used by the virtual character has an impact on users' judgments and employment decisions: when the virtual character uses an amused smile the users perceive the job as more positive and more suitable than when the virtual character exhibits a polite smile or a neutral expression. Note that the virtual character smiles when telling an amusing utterance, *i.e.* in a situation in which the user may expect an amused smile.

Moreover, as shown in Theonas, Hobbs and Rigas [31], smiles of virtual characters, expressed in an appropriate situation, enable the creation of a sense of comfort and warmth and a global friendly and living atmosphere.

In conclusion, when displayed by a human, the amused and the polite smile may be distinguished through morphological and dynamic characteristics. Despite some specific muscle contractions associated to smile types, no consensus exists in the literature on the facial characteristics of amused and polite smiles (Section 2.1.1). In the context of virtual characters, researchers have mainly focused on amused smile to express an emotion of joy, and sometimes on polite smile (in the particular context of a fake smile) to mask an expression of sadness. In our work, we propose a method to design virtual character's smiles that are directly created by users. We then explore the effects of these expressed smiles on the user's perception of the virtual character.

Research shows that smiles expressed both by a human or a virtual character enhance the social stance perceived by others, and particularly for smiling male (be virtual or human) and for a displayed amused smile. However, existing research has mainly compared the global perception of an agent (virtual or human) expressing no smile or an amused or a polite smile. In our work, we investigate the effect of a virtual character displaying both smiles at different moment of its speech.

Before presenting the study on the effect of smiles on user's perception, we first introduce the method used to characterize the features of virtual character's amused and polite smiles.

3. THE CHARACTERISTICS OF VIRTUAL SMILES

In order to identify the morphological and dynamic characteristics of the amused and the polite smile of a virtual character, we have proposed a human-centric approach: we have created a web application that enables a user to easily create different types of smile on a virtual character's face. Through radio buttons on an interface, the user could generate any smile by choosing a combination of seven parameters (amplitude of smile, duration of the smile, mouth opening, symmetry of the lip corner, lip press, and the velocity of the onset and offset of the smile). We have considered two or three discrete values for each of these parameters (for instance, small or large for the amplitude of the smile). These parameters were selected as being pertinent in smile behaviors [22]. When the user changes the value of one of the parameters, the corresponding video of a virtual character smiling is automatically played. Considering all the possible combinations of the discrete values of the parameters, we have created 192 different videos of smiling virtual character. The user was instructed to create one animation for each type of smile. Three hundred and forty eight participants (with 195 females) with a mean age of 30 years have created smiles. We have then collected 348 descriptions for each smile (amused and polite). The experiment is presented in details in [23].

Based on this smile corpus and on a decision tree classification technique, we have defined an algorithm to determine the morphological and dynamic characteristics of the smile types that a virtual character may express. We have chosen to use decision tree learning as this technique is well-adapted to qualitative data and produces results that are interpretable and that is easily implemented in a virtual character. By applying the CART (Classification And Regression Tree) method [3], with the morphological and dynamic characteristics as input variables and the types of smile as target variables, we have obtained a decision tree in which the *nodes* correspond to the smile characteristics and the *leaves* to the smile types. In the resulting decision tree, 10 leaves are labeled as polite smiles, and 7 as amused smiles. The advantage of such a method is to consider, not only one amused or polite smile but several smile types. That enables one to increase the repertoire of the virtual character's expressions. The global error rate is 27.75%, with a 95% confidence interval of 1.2%: the global error rate is in the interval [26.55%, 28.95%] (for more details on the corpora of smiles and the proposed algorithm, see [22]).

To validate the resulting smiles, an evaluation of four of the best classified amused and polite smiles have been performed in context. Different scenarios (of polite and amused

situation) were presented in text to the user. For each scenario, video clips of virtual character's different smiles were presented. We asked users to imagine the virtual character displaying the facial expression while it was in the situation presented in the scenarios. The user had to rate each of the facial expressions on its appropriateness for each given scenario. The evaluation has been conducted on the web through a platform of tests developed using Flash technology. Seventy-five individuals participated in this evaluation (57 female) with a mean age of 32. The evaluation revealed significant results showing that the generated smiles are appropriate to their corresponding context (for more details on the experiment, see [23])

The next step is to measure the effect of these smiles on partners of an interaction. For this purpose, we have conducted a study to identify how users perceive smiling virtual characters saying a sentence, varying the gender of the virtual character and the types of smile being expressed. We present in more details this study in the next section.

4. MEASURING THE EFFECTS OF SMILING VIRTUAL CHARACTERS

In order to measure the effects of the expressions of smile by virtual characters, based on a human-centric approach, we have conducted a study to collect perception the users have of a virtual character when the later displays polite and amused smiles. We consider the situation in which the virtual character expresses smiles when speaking[1]. Given the types of smile considered, we have chosen positive situations to match the types of smile. The agent tells a joke to the user. The display of an amused smile by the virtual character is relevant in this situation. The polite smile is used to accompany the virtual character's salutation at the beginning of its talk [24, 4].

Procedure.

We performed the evaluation on the web through a platform of tests developed using Flash technology. The test has two parts. In the first part, each participant watches four videos of a virtual character telling a joke (Figure 1): two video clips of a female virtual character telling a joke and two video clips of a male virtual character telling a joke. The four jokes told to the participant were different. To try to ensure that the user watched each video clips, we imposed that the user cannot go to the next page before clicking on the play button of the video clip. The order of the video clips has been counterbalanced to avoid an effect of the order on the results. In total the duration of the test was around 20 minutes.

After watching each video clip, the user had to rate the stance of the virtual character on a Likert scale of 5-points. Stance is defined in Scherer [28] as "affective style that spontaneously develops or is strategically employed in the interaction with a person or a group of persons, coloring the interpersonal exchange in that situation (e.g. being polite, distant, cold, warm, supportive, contemptuous)". In this study, we have considered the following stances as being relevant to the scenarios: *spontaneous*, *stiff*, *cold*, *warm*, *boring*, and *enjoyable*. Moreover, to measure the effect of smiles on the perception of what the agent said, we asked the user to indicate how well she understood the joke and if she likes it.

In the second part of the test, four videos of the virtual character smiling were presented to the user. Here, the virtual character just smiles without speaking. For each video, we asked the user to indicate the types of smile displayed by the virtual character: *polite*, *amused*, *none of them* (Figure 2). In this way, we verify if the smiles are perceived by the users as expected. Once again, the order of the presented videos was counterbalanced to avoid an effect of their order on the results.

Figure 2: Screenshot of the second part of the test

Smiles.

The video clips presented to the user correspond to the smiles resulting from our algorithm and that were validated by the evaluation (Section 3). For each type, we used two different smiles with a good recognition rate. Table 1 indicates the characteristics of these smiles.

id	type	size	mouth	sym.	lip	cheek	onset	dur.
1	pol.	small	close	yes	no	no	0.4s	3s
2	pol.	small	close	no	no	no	0.4s	1.6s
3	amu.	large	open	yes	no	yes	0.8s	3s
4	amu.	large	open	yes	no	yes	0.8s	1.6s

Table 1: The characteristics of the amused (amu) and polite (pol) smiles. In the first line, *size* indicates the size of the lip extension, *mouth* indicates if the mouth is opened or closed, *sym* indicates if the smile is symmetric or not, *lip* if the lip is pressed, *cheek* if the cheek is raised, *onset* the duration of the onset and offset, and *dur* the total duration of the smile.

Virtual characters.

In order to measure the effect of gender on the user's perception of virtual character's smiles, we have considered two different virtual characters: one female, named *Poppy*, and one male, named *Obadiah*. Figure 3 illustrates the virtual characters Poppy and Obadiah smiling.

Virtual characters' talk.

[1]We do not explore the display of smile when the virtual character is listening, *i.e.* smiles used as backchannel. For instance see [2] for a study on its effect on user's perception.

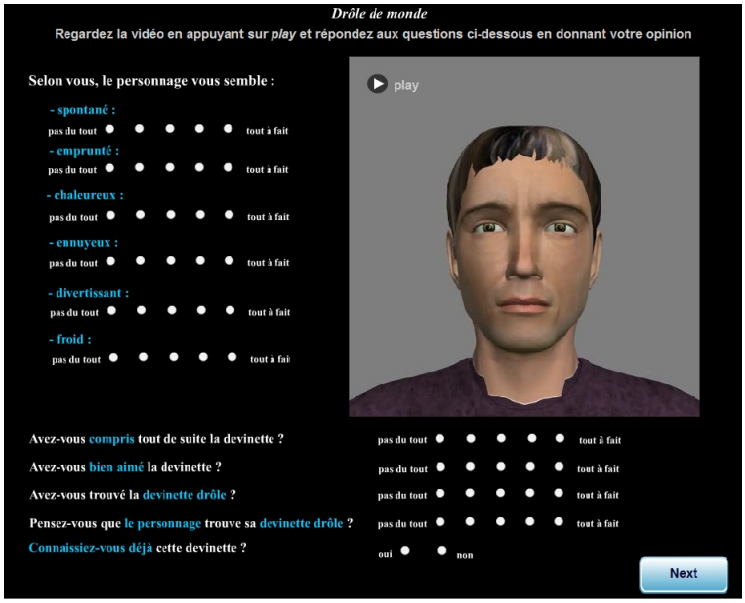


Figure 1: Screenshot of the first part of the test

Figure 3: Screenshot of the two virtual characters smiling

The virtual characters spoke French. The video clips presented to the participants correspond to the virtual characters telling a riddle to the user after a brief salutation. For instance (translated from French): "Good morning, I know a little riddle, what is the future of I yawn? I sleep! ". Four different riddles have been selected based on a brief evaluation of sixteen riddles. We have asked 7 persons (3 females and 4 males) to rate their liking of the sixteen riddles between 0 and 5. Based on the results, we have selected the riddles with the maximum rate and the minimum standard deviation. We suppose that the selected four riddles are approximatively equivalent. Finally, in terms of verbal behavior of the virtual character, only the riddle varies from one video clip to another. The beginning of the talk and the tonality of the voice do not vary.

Concerning the non-verbal behavior, only the smiles (both the type of smiles and the moment when it is expressed) differ from one video clip to another. Four conditions have been considered:

- *no smile condition*: the virtual character expresses no smile during its talk;
- *polite smile condition*: the virtual character displays only the polite smile when the virtual characters is saying "good morning";
- *amused smile condition*: the virtual character expresses only the amused smile when it says the response to the riddle;
- *both smiles condition*: the virtual character displays the polite and amused smiles at the moment described in the polite and amused conditions.

The different smiles expressed by the virtual characters are those described Table 1.

Participants.

Two hundred and forty two individuals participated in this study (158 female) with a mean age of 30 (SD = 10.35). They were recruited via mailing lists on line. The participants were mainly from France (N = 223), followed by Belgium (N = 5). There was some participants from Germany, Algeria, Tunisia, and Italy. Each participant has watched four video clips (two of Poppy and two of Obadiah telling each a different riddle)[2] and four video clips of the virtual characters just smiling (in the second part of the test).

In the next section, we present in details the results of this test.

5. USER'S PERCEPTION OF SMILING VIRTUAL CHARACTERS

5.1 Results

First of all, we have analyzed the results of the second part of the test to ensure that smiles have been perceived correctly, *i.e.* amused smiles have been tagged as amused and polite smiles as polite by the participants. Globally, the smiles have been in average categorized correctly, except one amused smile displayed by Poppy categorized in average

[2]Note that the experimental design does not correspond to repeated measures design because each participant is not exposed to all the conditions of the experiment.

more as polite than as amused smile (smile with the id 4 in Table 1). We have then decided to exclude the video clips in which Poppy displays this smile. In total, we have considered 483 video clip's rating.

To measure the effects of smiles on the user's perception, we have performed ANOVAs and the post hoc Tukey's test to evaluate the significant differences of rating between the different conditions (no smile, polite smile, amused smile and both smiles condition).

The significative results are presented in Tables 2. The first column indicates the condition compared (N for no smile, A for amused smile, P for polite smile, and AP for both smiles condition) and the first line the studied social stance. The elements of the table correspond to the condition in which the social stance of the virtual character has been the best perceived (n.s. means non significant, *: $p < .05$, **: $p < .01$, ***: $p < .001$). For instance, in Table 2, the notation A^* at the intersection of the line *N-A* and the column *Enjoyable* means that, in the amused smile condition, the virtual character has been perceived significantly more enjoyable (with $p < .05$) than in the no smile condition.

	Warm	Enjoyable	Cold	Boring
N-A	A^{***}	A^{*}	N^{*}	n.s.
N-P	P^{***}	n.s.	n.s.	n.s.
N-AP	AP^{***}	AP^{**}	N^{***}	N^{**}
P-AP	AP^{***}	AP^{*}	n.s.	n.s.

Table 2: Comparison of the user's perception of the virtual character's social stance in the different conditions

To measure the effects of gender, we have performed T-Test. The gender of the virtual characters has significant effects on the user's perception. For instance, when Poppy is smiling (whatever is the smile), she is perceived significantly less *cold* (and warmer) than Obadiah expressing the same smile (with $p < 0.05$). Poppy is perceived less *boring* with one smile (polite or amused) than Obadiah with the same smile (with $p < 0.05$). With the amused smile (with or without a polite smile), Poppy is perceived significantly more *spontaneous* and *enjoyable* than Obadiah expressing the same smile (with $p < 0,01$).

With regard to these results, we have more precisely analyzed the significant differences for each virtual character separately. Contrary to the results presented in Table 2, it appears that, compared to the expression of only the polite smile, Poppy is perceived significantly more *spontaneous*, *warm* (and less *cold*), and less *stiff* when it expresses an amused smile (with $p < 0.05$). For Obadiah, the expression of an amused smile (with or without a polite smile) enhances the warm impression of the virtual character (with $p < 0.05$).

Concerning the effect of the gender of the user on her perception, only one significant result has been noticed: women perceive the virtual character as significantly more polite when it expresses a polite smile than men. This result can be explained in the light of the research of [15] showing that women are more sensitive to non-verbal behaviors and more able to decode facial expressions cues for virtual characters' faces (Krumhuber et al., 2007), and of the research of [13] showing that women make more extreme judgment ratings than men when decoding facial expressions.

Concerning the effects of smile on the perception of the content of the sentence, significant differences appear. The users prefer the riddle and judge the riddle funnier when the virtual character expresses both smiles than with no smile or only one (polite or amused) (with $p < 0.05$). Moreover, as expected, the user judges that the virtual character thinks its riddle funnier when it expresses an amused smile (with or without a polite smile) compared to the expression of no smile or only a polite smile (with $p < 0.001$). This result confirms that the amused smile is viewed by the user as an information on the positive state of the virtual character.

We discuss in more details the results of the study in the next section.

5.2 Discussion

The results of the study confirm that smiles enhance the social stance of a virtual character. Indeed, globally, the smiles (both the polite and amused smiles) increase the warm stance of the character. Particularly, the amused smile enables to improve the perception of the virtual character in terms of enjoyment compared to no smile or a polite smile. The display of the polite and the amused smile in the same sentence enables to decrease the boring stance of the virtual character. These results are consistent with previous research showing that individuals and virtual entities who smile are perceived more positively than non-smiling agents (see Section 2.2). However, the results also highlight the impact of the different smiles on the user's perception, showing that the display of an amused smile enables one to enhance certain social stances of the virtual character (warm and enjoyment) compared to the display of a polite smile. These results can be explained as amused smile is commonly associated to felt smile reflecting a positive emotion, compared to polite smile generally associated to fake smile. These effects on the perception of the virtual character's stances confirm that the users perceive the difference between smiles, and more particularly between their associated communicative intention, when the virtual character displays them in a talk. The results show that the use of both smiles enables one to decrease the boring stance of the character. That can be due to the variability of smiles expressed by the virtual character in appropriate situation. It may reflect more engagement from the virtual character.

A gender effect was also revealed. The female virtual character displaying an amused smile is perceived more positively (spontaneous, warm, enjoyable) that the male virtual character expressing the same smile. In particular, it seems that to add an amused smile in a sentence with a polite smile enables one to decrease the stiff stance. In contrary, the male virtual character is generally perceived more boring and cold when smiling (whatever is the smile) compared to a smiling female virtual character. Whereas previous research in Human and Social Science has shown that the absence of smile for a woman deteriorates her image compared to a man (see Section 2.2), the results of our study show that the smile displays by a female virtual character enables it to enhance her image (spontaneous, warm, enjoyable) compared to a male virtual character. These results confirm the recent experiment reported in Kulms, Krämer, Gratch, and Kang [16] showing that virtual character's non-verbal behavior may be predominant on stereotype attribution.

In the next section, based on the results of the experiment,

we attempt to propose a model to automatically compute how the (potential) perception of the user of the virtual character's stance[3] evolves depending on its smiling behavior.

5.3 Toward a model of user's perception of a smiling virtual character

In order to enable a virtual character to approximate the user's perception of its social stance, we propose a first model to automatically compute the potential perception of the user depending on the smiles displayed by the virtual character when speaking. This model aims at estimating the probability that a virtual character is perceived as *spontaneous*, *stiff*, *warm*, *enjoyable* and *boring*. In the collected data on user's perception (Section 4), each stance was rated along a 5 point Likert scale, we can represent this as natural values ranging from 0 to 4. To provide convenient and intelligible variables, we map the discrete values to three categories: the value 0 is associated to *neutral*, the two lowest values (for x=1 or x=2) are associated to *low*, and the two highest values (for x=3 or x=4) are associated to *high*. The probabilities to obtain such values for each social stance are computed based on the results of the study (Section 5.1). For instance, the probability that the female virtual character is perceived highly spontaneous by displaying an amused smile when telling something positive is $P(spontaneous = high|(smile = A \vee smile = AP) \wedge gender = female) = 0.27$, *i.e.* the probability that spontaneous=3 or spontaneous=4 in the condition A (only an amused smile is expressed) or AP (an amused and polite smile are expressed). Finally, after each sentence is pronounced by a virtual character, given its gender and its smiling behavior (polite smile, amused smile, both smiles, or no smile), the model provides a matrix reflecting the probability of the user's (potential) perception of the virtual character's social stance. For instance, the matrix illustrated in Figure 4 reflects the potential social stance perceived by the user for a male virtual character which has not expressed a smile when telling something positive.

	neutral	*low*	*high*
spontaneous	0.44	**0.47**	0.09
stiff	0.14	**0.49**	0.37
warm	**0.54**	0.42	0.04
enjoyable	0.37	**0.55**	0.09
boring	0.06	0.29	**0.65**
cold	0.05	0.28	**0.67**

Figure 4: Matrix of probabilities representing the user's (potential) perception of a male virtual character that does not display an amused smile when telling a riddle.

The model enables us to measure the effects of smile but also the effect of *not* displaying a specific smile in a situation in which the user may expect this non-verbal behavior.

Thus, it is an attempt to compute how user's (potential) perception of its interactant's social stance, based on its nonverbal behavior, evolves during an interaction. The proposed approach is *human-centric* since both the signals themselves, their corresponding communicative functions, and their impacts on the perceptive social stances, have been defined by the users. The resulting model characterizing the social stances that the user attributes to the virtual character given its smiling behavior, can be viewed as a model of the user's Theory of Mind (ToM, [5]) on the social stance inferences.

Our model still needs to be extended in several directions. It has been constructed from results emanating from a specific scenario (saying riddle). We still have to see if it still hold for situations in which the virtual character does not only tell something funny (like a riddle), but something globally positive (*i.e.* reflecting a positive emotion). Similar concerns hold for the perception of polite smile. We need to validate if we can extend our model to any situation for which the expression or non-expression of a polite smile is expected. Greeting is such a situation but there are others as those described in Ochs, Niewiadomski, Brunet, and Pelachaud [23]. Moreover we suppose that the computed probabilities are cumulative during the interaction. For instance, several successive sentences reflecting a positive emotion without displaying an amused smile will lead to successive decreasing of the user's perception of the virtual character's positive social stance. This hypothesis has to be validated during virtual character-user interaction.

6. CONCLUSION

In conclusion, in this paper, we have performed a study to measure the effects of virtual characters displaying polite and amused smiles when saying a sentence, on the user's perception of the virtual character's social stance. The results of the study have revealed significant differences, confirming that smiles enhance the social stance of a virtual character. These results are consistent with previous research showing that individuals and virtual entities who smile are perceived more positively than non-smiling agents (Section 2.2). Moreover, the results also highlight the impact of the different smiles on the user's perception, showing that the display of an amused smile enables one to enhance certain social stances of the virtual character (warm and enjoyment) compared to the display of a polite smile. In our experiment we have considered the expression by a virtual character of both polite and amused smiles when speaking. Previous research has mainly studied the effects of these smiles separately whereas in communication both smiles are generally expressed.

Our results also provide new insights concerning the gender effect on the user's perception. Indeed, contrary to human-human interaction, a smiling female virtual character seems to be better perceived than a smiling male virtual character.

Based on the measures collected during the study (Section 5.1), a probabilistic model of the user's (potential) perception of a smiling virtual character has been proposed. It enables one to evaluate the user's perception of virtual character's social stance during the interaction given the virtual character's gender and its smiling behavior. Both the effect of the expression of smile in appropriate situations and the absence of smile in expected smiling situations have been modeled.

The next step consists in evaluating such a model during an interaction with users. For this purpose, we aim

[3]We do not model the user's perception of the virtual character's speech content since our results are closely linked to the specific context of the talk.

at integrating our model in the platform SEMAINE [29] to test at several moments during the interaction if the proposed model provides an adequate virtual character's image of what users have.

7. ACKNOWLEDGMENTS

This research has been supported by the European Community Seventh Framework Program (FP7/2007-2013), under grant agreement no. 231287 (SSPNet).

8. REFERENCES

[1] Z. Ambadar, J. F. Cohn, and L. I. Reed. All Smiles are Not Created Equal: Morphology and Timing of Smiles Perceived as Amused, Polite, and Embarrassed/Nervous. *Journal of Nonverbal Behavior*, 17-34:238–252, 2009.

[2] E. Bevacqua, S. Hyniewska, and C. Pelachaud. Positive influence of smile backchannels in ECAs. In *International Workshop on Interacting with ECAs as Virtual Characters (AAMAS)*, 2010.

[3] L. Breiman, J. Friedman, R. Olsen, and C. Stone. *Classification and Regression Trees*. Chapman and Hall, 1984.

[4] J. Cassell, T. Bickmore, L. Campbell, H. Vilhjálmsson, and H. Yan. More than just a pretty face: Conversational protocols and the affordances of embodiment. *Knowledgebased Systems*, 14(1-2):55–64, 2001.

[5] D. C. Dennett. *The Intentional Stance*. Mit press edition, 1987.

[6] F.M. Deutsch, D. LeBaron, and M.M. Fryer. What is in the Smile? *Psychology of Women Quarterly*, 11, 1987.

[7] J.A. Edinger and M.L. Patterson. Nonverbal involvement and social control. *Psychological Bulletin*, 93:30–56, 1983.

[8] P. Ekman and W. V. Friesen. *Unmasking the Face. A guide to recognizing emotions from facial clues*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1975.

[9] P. Ekman and W. V. Friesen. Felt, False, And Miserable Smiles. *Journal of Nonverbal Behavior*, 6:238–252, 1982.

[10] P. Ekman, W. V. Friesen, and J. C. Hager. *The facial action coding system*. Weidenfeld and Nicolson, 2002.

[11] M. G. Frank, P. Ekman, and W. V. Friesen. Behavioral markers and recognizability of the smile of enjoyment. *Journal of Personality and Social Psychology*, 64:83–93, 1993.

[12] U. Hess, S. Blairy, and R. E. Kleck. The influence of facial emotion displays, gender, and ethnicity on judgments of dominance and affiliation. *Journal of Nonverbal Behavior*, 24:275–283, 2000.

[13] M. Katsikitis, I. Pilowsky, and J. M. Innes. Encoding and decoding of facial expression. *Journal of General Psychology*, 124:357–370, 1997.

[14] E. Krumhuber, A. Manstead, D. Cosker, D. Marshall, and P. Rosin. Effects of Dynamic Attributes of Smiles in Human and Synthetic Faces: A Simulated Job Interview Setting. *Journal of Nonverbal Behavior*, 33:1–15, 2008.

[15] E. Krumhuber, A. Manstead, and A. Kappas. Temporal aspects of facial displays in person and expression perception. The effects of smile dynamics, headtilt and gender. *Journal of Nonverbal Behavior*, 31:39–56, 2007.

[16] P. Kulms, N. C. Krämer, J. Gratch, and S. Kang. It's in Their Eyes: A Study on Female and Male Virtual Humans' Gaze. In *Intelligent Virtual Agent (IVA)*, pages 80–92, 2011.

[17] M. LaFrance and M. A. Hecht. Why smiles generate leniency. *Personality and Social Psychology Bulletin*, 21(3):207–214, 1995.

[18] S. Lau. The effect of smiling on person perception. *Journal of Social Psychology*, 117:63–67, 1982.

[19] M. Mehu, A.C. Little, and R.I.M. Dunbar. Duchenne smiles and the perception of generosity and sociability in faces. *Journal of Evolutionary Psychology*, 5(1-4):133–146, 2007.

[20] M. M. Moore. Nonverbal courtship patterns in women. *Ethology and Sociobiology*, 6:237–247, 1985.

[21] R. Niewiadomski and C. Pelachaud. Model of Facial Expressions Management for an Embodied Conversational Agent. In *2nd International Conference on Affective Computing and Intelligent Interaction (ACII)*, pages 12–23, Lisbon, Portugal, 2007.

[22] M. Ochs, R. Niewiadmoski, and C. Pelachaud. How a virtual agent should smile? morphological and dynamic characteristics of virtual agent's smiles. In *Intelligent Virtual Agent (IVA)*, 2010.

[23] M. Ochs, R. Niewiadomski, P. Brunet, and C. Pelachaud. Smiling virtual agent in social context. *Cognitive Processing, Special Issue on"Social Agents"*, 2011.

[24] I. Poggi and R. Chirico. The meaning of smile. In *Oralite, gestualite, communication multimodale, interaction*, pages 159–164. 1998.

[25] I. Poggi and C. Pelachaud. *Affective Interactions: Towards a New Generation of Computer Interfaces*, chapter Emotional. 2000.

[26] M. Rehm and E. André. Catch me if you can ? Exploring lying agents in social settings. In *AAMAS*, pages 937–944. Academic Press Inc, 2005.

[27] H. T. Reis, W.I. McDougal, C. Monestere, S. Bernstein, K. Clark, E. Seidl, M. Franco, E. Giodioso, L. Freeman, and K. Radoane. What is smiling is beautiful and good. *European Journal of Social Psychology*, 20:259–267, 1990.

[28] K. R. Scherer. What are emotions? And how can they be measured? *Social Science Information*, 44(4):695–729, December 2005.

[29] M. Schröder. The SEMAINE API: Towards a Standards-Based Framework for Building Emotion-Oriented Systems. *Advances in Human-Computer Interaction*, 2010.

[30] E. Tanguy. *Emotions: the art of communication applied to virtual actors*. PhD thesis, Department of Computer Science, University of Bath, England, 2006.

[31] G Theonas, D Hobbs, and D Rigas. Employing Virtual Lecturers' Facial Expressions in Virtual Educational Environments. *International Journal of Virtual Reality*, 7:31–44, 2008.

Session 3A
Robotics I

Supervised Morphogenesis – Morphology Control of Ground-based Self-Assembling Robots by Aerial Robots

Nithin Mathews, Alessandro Stranieri, Alexander Scheidler, Marco Dorigo
IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium
{nmathews,astranie,ascheidler,mdorigo}@ulb.ac.be

ABSTRACT

In this paper, we study a heterogeneous robot team composed of self-assembling robots and aerial robots that cooperate with each other to carry out global tasks. We introduce *supervised morphogenesis* – an approach in which aerial robots exploit their better view of the environment to detect tasks on the ground that require self-assembly, and perform on-board simulations to determine the morphology most adequate to carry out the task. In case existing morphologies on the ground do not match those determined in simulation, aerial robots use a series of enabling mechanisms to initiate and control (hence supervise) the formation of morphologies more adequate to carry out the task. Supervised morphogenesis solely employs LEDs and camera-based local communication between the two robot types. We validate the applicability of our approach in a real-world scenario, in which ground-based robots are given the task to cross an unknown, undulated terrain by forming ad-hoc morphologies under the supervision of an aerial robot.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Robot teams, multi-robot systems, air/ground systems, self-assembling robots, swarm robotics

1. INTRODUCTION

Self-assembling robotic systems have been the topic of many studies (refer to [1] for an overview). In such systems, autonomous robots form new or re-arrange existing physical connections to each other to form distinctive collective robot structures (hereafter called morphologies). This morphological flexibility gives self-assembling robots the potential to adapt to changing environmental conditions. For instance, navigating through an uneven terrain may require a morphology different from the one required for pushing an object. However, existing systems are often not able to adaptively form new morphologies as a function of the task or the environment. This is primarily because individual components in existing systems consist of rather simple ground-based robots that are adversely affected by obstructed sensor views. Additionally, self-assembling robots often do not have the sensory apparatus required to determine the morphological constraints imposed by their environments.

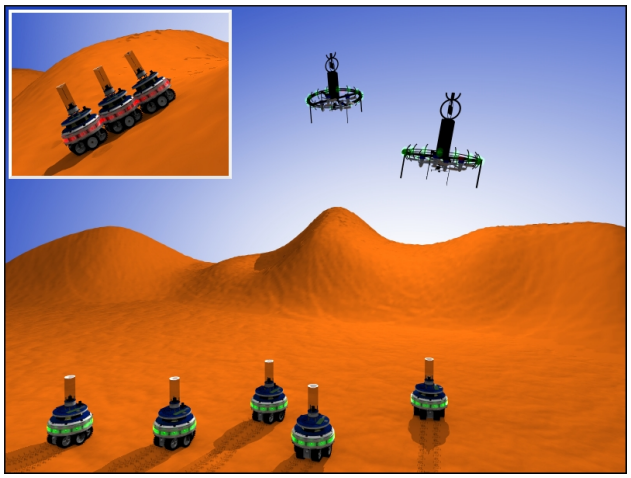


Figure 1: A potential deployment of the heterogeneous robot team considered in this study. Maneuvers in such undulated terrain may require the self-assembling robots to rely on environmental perception from an aerial perspective to determine the shape and the size of the morphologies (a chain morphology of size three is shown in the inset) that may allow the robots to navigate through the environment.

Although many algorithms have been proposed to control morphology formation in self-assembling robots [2–6], little attention has been devoted to the subsequent practical applicability of self-assembled morphologies. In fact, only very few works have considered real-world tasks that require robots to form task-dependent morphologies of precise shapes and sizes (hereafter called *target* morphologies) [7,8]. Due to sensory limitations, however, the robots in these works are not able to detect the tasks allocated to them. Therefore, the target morphologies were either predefined [7] or controlled through additional environmental cues [8]. In [9], self-assembling robots are able to detect and solve a series

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

of tasks. However, the considered tasks did not require a precise morphology shape or size to be solved successfully.

Many researchers have proposed to overcome the sensory limitations of ground-based robots by using heterogeneous systems composed of both ground-based and aerial robots [10–15]. These systems exploit the complementary capabilities of the two robot types. In fact, while aerial robots can offer unobstructed field of view and rapid coverage of large areas, ground-based robots can offer high accuracy sensing at relatively short distances and can manipulate the environment. In [10, 11], researchers have proposed GPS-based solutions to study localization and navigation problems in robot teams composed of aerial and ground-based robots. The robots in [12] complement each others observations by fusing sensory data to provide scalable solutions to tasks involving searching and tracking of ground targets. Other studies include cooperative surveillance [13, 14] and motion control of ground-based robots through an aerial robot [15]. Despite the variety of tasks and applications considered in existing air and ground-based robotic systems, no research has been carried out, to the best of our knowledge, to study how aerial robots may assist ground-based self-assembling robots in their morphogenetic processes.

In this paper, we enhance the sensing capabilities of a ground-based self-assembling robotic system by integrating aerial robots into the system (Fig. 1 shows a possible deployment scenario for such a heterogeneous team of robots). We propose *supervised morphogenesis* – an approach that enables aerial robots i) to detect tasks on the ground that require ground-based robots to self-assemble, and ii) if necessary, to initiate and control (i.e., supervise) the formation of appropriate morphologies. That is, ground-based robots delegate the decision-making concerning *if* and *what* morphologies to form to the aerial robots. The aerial robots exploit their elevated position and their richer sensory equipment to determine exact characteristics of tasks. Subsequently, they use *on-board simulations* to determine an appropriate target morphology. In particular, the aerial robots build a model of the perceived environment and then simulate the behavior of different morphologies within this environment. In this manner, aerial robots can assess how different morphologies perform when executing a task without requiring any physical realization of morphologies on the ground. Depending on the outcome of the simulations, aerial robots then determine the most appropriate target morphology and supervise its formation. Such a system has the ability to adapt to completely unknown environments and thus to significantly increase its level of autonomy.

We present the results of a first implementation of supervised morphogenesis. We report on experiments conducted to evaluate our approach in a real-world hill-climbing task. In this task, a group of ground-based robots and an aerial robot are required to reach a light source by navigating over a hill of unknown steepness. In our approach, the aerial robot calculates a height map using stereo images to build an internal representation of the perceived environment. They then execute on-board simulations to estimate the steepness each ground-based robot may experience when navigating to the light source. In case the simulations predict a ground-based robot to topple over, because of a too steep slope, the aerial robot positions itself over the hill to supervise the formation of target morphologies that guarantee safe crossing of the hill. In this initial implementation of supervised morphogenesis, we restrict target morphology to chain morphologies[1] composed of either two or three ground-based robots.

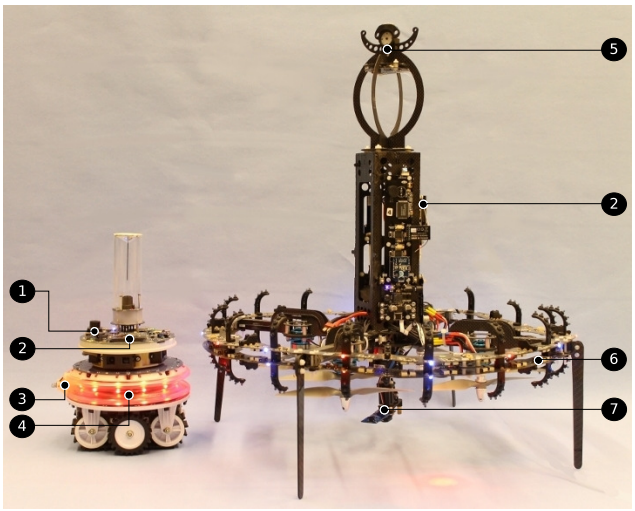


Figure 2: The two robot types considered in this study. The foot-bot is shown on the left while the eye-bot is on the right. 1) The upward-pointing camera, 2) the ARM11™processor, 3) the docking unit, 4) the docking ring with integrated LEDs, 5) the ceiling attachment device, 6) the downward-facing LED ring, and 7) the downward-pointing camera.

2. HARDWARE PLATFORM

In our experiments, we use a set of self-assembling robots called *foot-bots* and a flying robot called *eye-bot* (see Fig. 2). Both robot types were developed as part of the SWARMANOID project [16].

A foot-bot is a mobile robot with a circular chassis of 17 cm diameter. A combination of tracks and wheels provides the foot-bots with differential drive motion capabilities. The docking module provides self-assembling capabilities with other foot-bots. This module is composed of a docking unit with three fingers, a docking ring, and an integrated force sensor that can register the forces applied to the unit. A foot-bot can physically attach to another foot-bot by inserting the docking unit into its docking ring and then opening the three fingers. A foot-bot is also equipped with 12 RGB LEDs distributed around its docking ring. The LEDs allow a foot-bot to visually display its internal state to nearby robots. Other features include a 2D distance scanner, 24 IR proximity and light sensors, one upward-pointing and one omnidirectional 2 mega pixel HD camera supporting high quality vision in both vertical and horizontal planes. A custom-made on-board device named mxRAB can be used to exchange messages (10 bytes) and to estimate the relative range and bearing (up to a distance of 5 m) between adjacent foot-bots. This device combines radio frequency and infrared and is based on the work presented in [17].

An eye-bot is 54 cm high and has a diameter of 50 cm. Eight rotors, mounted in a co-axial quadrotor configuration, provide the eye-bot with thrust and control. The eye-bot

[1]A linear structure in which each robot besides the first one is connected to the rear of the preceding robot.

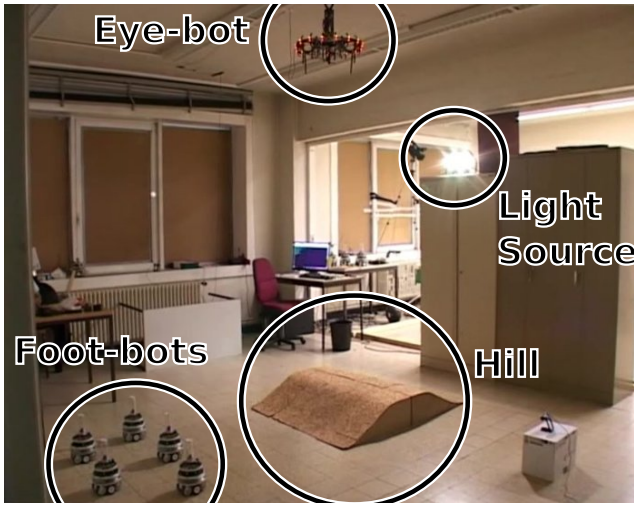


Figure 3: The experimental setup: 5 foot-bots, an eye-bot attached to the ceiling, a light source, and a mock-up hill.

has an on-board battery that allows 10-20 minutes of autonomous flight. It is also equipped with a ceiling attachment device that can be used in indoor environments to extend mission endurance. A downward-pointing 2 mega pixel HD 360° pan-and-tilt camera allows the eye-bot to survey the ground and to detect the foot-bots. The downward facing RGB LED ring with 16 RGB LEDs can be used to communicate internal state information to the foot-bots. Other features include a light weight body (270 g) made out of carbon-fiber, a 3D relative positioning sensor (with a maximum range of 12 m), an altitude sensor, and a magnetometer to detect heading direction.

Foot-bots and eye-bots are equipped with an on-board ARM11™processor (i.MX31 operating at 533 MHz with 128 MB RAM) running a Linux-based operating system that is interfaced with all on-board sensors and actuators.

3. TASK AND EXPERIMENTAL SETUP

A group consisting of 5 foot-bots and one eye-bot is given the task to navigate from a deployment area to a light source by crossing a hill of a priori unknown steepness (see Fig. 3). The inclination of the hill can vary between 0° (i.e., no inclination) and 30°. Individual foot-bots are only able to withstand a maximum inclination of 25° without toppling over. If for a hill the maximum inclination is less than 25°, individual foot-bots can cross without requiring further assistance. If, on the other hand, the maximum inclination is higher than 25°, the foot-bots have to self-assemble into chain morphologies that offer sufficient stability when passing over the hill. The number of chain morphologies that have to be formed and their individual sizes depend on the total number of foot-bots allocated to the task and are not known to any of the robots. In our experiments, the eye-bot is assumed to have flown in advance and attached to the ceiling[2] at a height of 2.96 m immediately before the hill. The task is considered accomplished if all 5 foot-bots manage to reach the light source.

[2]As it is irrelevant to the work presented in this paper and because it goes beyond the scope of this work, we do not discuss flight control algorithms that may result in this behavior of the eye-bot.

4. METHODOLOGY

We describe the methodology employed to solve the task considered in this work. First, the eye-bot uses its downward-pointing camera from an elevated position to build an internal representation of the environment. In particular, two sequentially taken images from two distinct positions in the environment are used to compute a height map of the environment in the field of view (details are given in Sect. 4.1). Second, this height map is used to calculate height profiles along each foot-bot's estimated trajectory to the light source. Subsequently, on-board simulations are performed to estimate whether each foot-bot is able to drive over the computed height profile of its estimated trajectory without toppling over (see Sect. 4.2). In case the simulation predicts that a foot-bot would topple over, the eye-bot supervises the formation of target morphologies that offer the physical stability required to cross the hill. To initiate morphology formation, the eye-bot selects a favorably situated foot-bot. The eye-bot then establishes a dedicated one-to-one communication link with the selected foot-bot (see Sect. 4.3). The dedicated communication link is then used to initiate the formation of a target morphology by activating the execution of a SWARMORPH-script [2]. SWARMORPH-script is a language that permits arbitrary morphology generation using self-assembling robots in a distributed manner. The foot-bots are pre-loaded with two different SWARMORPH-scripts that, when executed, can generate a chain morphology composed of two or three foot-bots each.[3] Physical connections between a connection inviting foot-bot and a neighboring foot-bot are formed using the recruitment and guidance based mechanism presented in [18]. In Sect. 4.4, we finally present robot controllers we have developed while following a distributed control paradigm.

4.1 Internal representation of the environment

The eye-bot builds an internal representation of the ground underneath by computing a height map. Most flying robots are subject to payload limitations that reduce the possibilities for dedicated, on-board sensing hardware (such as Microsoft's Kinect) capable of computing height maps. In this section, we describe how the eye-bot obtains the height map using its comparatively lightweight monocular vision system.

The eye-bot takes two images (each from a different position) such that the closest foot-bot to the light source is always in the field of view. Based on the two images, the eye-bot computes the height of the surfaces and the objects in the scene. The extraction of three-dimensional information of a scene based on stereo images is a problem that has been studied by the computer vision community for decades [19].

We make a series of assumptions when acquiring the images. First, we assume that the eye-bot is able to hover above the ground at a fixed height using its altitude sensor and that the image plane is parallel to the ground. Second, the exact distance of the eye-bot to the ground and the distance between the two positions is assumed to be accessible to the eye-bot through its 3D relative positioning sensor. Third, we assume to know the focal length of the camera, obtained through a prior calibration step [20].

Both images are taken at a resolution of 640x480 pixels (see Fig. 4a and Fig. 4b). In order to compute a height map,

[3]In our experiments, the eye-bot makes the simplifying assumption that chain morphologies provide the physical stability required to cross any detected hill.

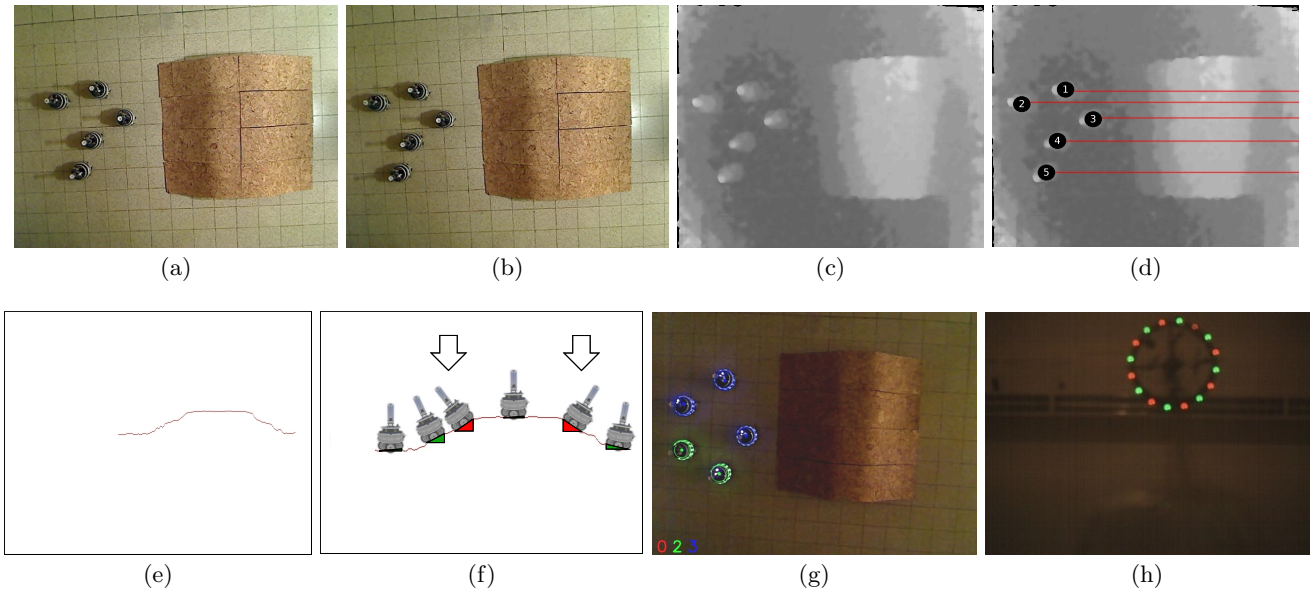


Figure 4: (a,b) Images acquired by the eye-bot at a distance of 30 cm from each other. (c) A grayscale representation of the disparity map computed. (d) The trajectory estimations of five foot-bots to the light source. (e) The height profile of the trajectory estimated for foot-bot #3. (f) A schematic of an on-board simulation run. The inclination computed at six different positions is shown. The arrows depict the location of computed inclinations that are higher than 25 °. (g) The eye-bot detects two GREEN and three BLUE signals. (h) A foot-bot detects the signal RED/GREEN sent by the eye-bot.

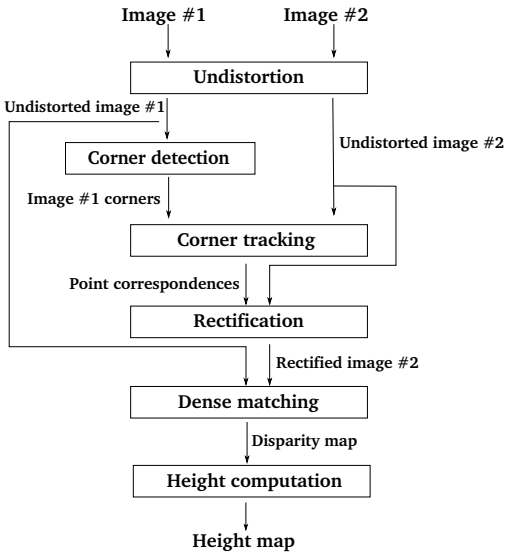


Figure 5: A flowchart depicting the individual steps included in the computation of the height map.

we first generate a so-called *disparity map*. For each point in the first image, a disparity map contains the distance (in pixel) by which the point has moved in the second image. Points further away from the camera move less compared to points closer to the camera. Based on the disparity of a point, the knowledge about the distance of the eye-bot to the ground, the displacement of the eye-bot between the positions from which the two images have been acquired, and the exact properties of the camera, the eye-bot can calculate the height of the point in real-world distances.

Fig. 5 shows a flowchart containing the individual steps of the process that leads to the computation of a height map. The *Undistortion* step takes both images as input and compensates for tangential and radial distortion introduced by the lens and modeled by the coefficients found by the calibration step. The next three steps transform the images so that the search for the correspondence of a point in the first image can be limited to a horizontal scan in the second image. This process is called *rectification*. First, the *Corner Detection* step finds interesting feature points in the first image as described in [21]. Then, in the *Corner Tracking* step, the iterative Lucas-Kanade method [22] is used to track the interesting points in the second image. The output of this step is a set of sparsely matched points. In the *Rectification* step, this set of correspondences is used to find the transformation matrix that vertically aligns the second image to the first. The transformation matrix is used to rectify the second image. To derive the disparity of each point in the image we then apply the *Dense matching* step. The algorithm used here is described in [23]. Figure 4c shows a grayscale representation of the disparity map. Finally, the last step applies stereo triangulation to each disparity value to produce the height map. The computed height map is a two-dimensional matrix of size 640x480. For each real-world point visible in both images, the height map contains its elevation from the ground in *cm*.

4.2 On-board simulation-based reasoning

Here we describe how the eye-bot uses the height map to perform on-board simulations. Based on these simulations, the eye-bot evaluates whether or not individual foot-bots are able to continue their navigation towards the light source without self-assembling.

The task considered in this work requires each foot-bot to

execute a phototaxis behavior that guides the foot-bot directly to the light source. Therefore, as shown in Fig 4d, the eye-bot estimates a foot-bot's trajectory to the light source to be a straight line. Note that while the position of a foot-bot in the image returned by the camera is calculated using the computer vision algorithms presented in Sect. 4.3, the light source is assumed to be situated on the right edge of the image. For each foot-bot, the eye-bot reads out the cell values in the height map on a line (of 1 pixel width) that connects the position of the foot-bot to the right edge of the image horizontally. These values represent the height profile of the foot-bot's estimated trajectory to the light source. An example of a height profile is plotted in Fig. 4e.

For each computed depth map, the eye-bot simulates a virtual navigation of each foot-bot in the field of view to the light source along its respective height profile. Each foot-bot is placed at its current position on the height profile and the inclination experienced by the foot-bot in this position is calculated. In particular, the inclination of the surface underneath the front and the rear end of the simulated foot-bot's chassis is calculated. Then, the foot-bot is moved pixel-by-pixel, until the foot-bot's chassis reaches the light source, and the inclination is calculated each time the foot-bot is moved (see Fig. 4f for a visualization). In case a calculated inclination for a foot-bot is higher than 25 °, the inclination a foot-bot can endure without toppling over, the eye-bot halts performing simulations and requires the foot-bots to self-assemble.

4.3 LEDs and camera-based communication

The robots considered in this work use their on-board LEDs and cameras to communicate with each other. The eye-bot uses its downward-pointing camera to perceive the LEDs of the foot-bots (see Fig. 4g). A foot-bot, in turn, uses its upward-pointing camera to perceive the LEDs of an eye-bot (see Fig. 4h).

A foot-bot can transmit three distinctive signals to the eye-bot by displaying either one of the three primary RGB colors on its LED ring (i.e., RED, GREEN, or BLUE). The eye-bot executes the following two steps to perceive the signal transmitted by each foot-bot and detect the total number of foot-bots in the field of view. First, the eye-bot carries out a threshold-based RGB color detection on the images returned by its downward-pointing camera. Second, for each RGB color channel, a circle detection algorithm is run to determine the number of foot-bots in the field of view. Each detected circle is assumed to be a foot-bot with its unique position (i.e., the center of the circle) in the environment.

In addition to the three signals used by the foot-bots, the eye-bot uses the following two signals to communicate to the foot-bots: RED/BLUE and RED/GREEN. These signals are based on two primary RGB colors displayed simultaneously using alternating LEDs. A foot-bot processes the image returned by its upward-pointing camera to detect RGB color blobs. The signal transmitted by the eye-bot is then determined by evaluating the total number of detected red, green and blue blobs respectively. For instance, if only red blobs are detected, the signal transmitted by the eye-bot is detected as RED. If, on the other hand, both red and green blobs are detected simultaneously the signal transmitted by the eye-bot is detected as RED/GREEN (see Fig. 4h).

In self-assembling robotic systems, morphology formation is usually initiated by a single robot. We use the mechanism presented in [24] to let the eye-bot select a favorably located foot-bot to initiate morphology formation by establishing a dedicated communication link to the foot-bot. The eye-bot uses the signals RED, GREEN, and BLUE in combination with an iterative selection process to narrow down the number of potential recipients of a broadcast signal (i.e., a color displayed on the LEDs) to a single foot-bot. In [24], this iterative selection process is shown to scale well with respect to the number of participating foot-bots. The established communication link with a particular foot-bot enables the eye-bot to ensure that a subsequently transmitted signal (for instance RED/GREEN or RED/BLUE) will only be processed by the selected foot-bot even though other foot-bots may also be able to receive the broadcasted signal.

4.4 Distributed robot control

We present two behavior-based controllers we have developed: one for the eye-bot and one for the foot-bots. Each robot is autonomous and independently executes its respective controller on the on-board ARM11™processor.

The behavior-based controller of the eye-bot is described by the finite state machine shown in Fig. 6a. Initially, the eye-bot executes a phototaxis behavior by flying ahead of the foot-bots towards the direction of the light source. The light source is detected using the downward-pointing pan-and-tilt camera. Simultaneously, the eye-bot estimates the distance traveled by using the 3D relative positioning sensor in combination with at least one stationary robot (for instance in the deployment area) that provides a static reference point. At fixed distance intervals, the eye-bot takes images of the ground.[4] Sequentially taken images are then used to compute a height map of the surface in the field of view. If subsequently performed on-board simulations do not predict danger in the surveyed area (i.e., foot-bots can act independently), the eye-bot continues heading towards the light source by executing the phototaxis behavior. If, on the other hand, simulations detect a surface too steep for individual foot-bots, the eye-bot positions itself (by attaching to the ceiling in indoor environments or otherwise by hovering) above the hazardous area. The eye-bot sends the signal RED to issue a warning to the foot-bots underneath.

From its elevated position, the eye-bot uses the signals RED, GREEN, and BLUE to establish a one-to-one communication link to a favorably located foot-bot that will initiate a target morphology formation. Each foot-bot that acknowledges the warning using the BLUE signal is a selection candidate. Among these, the eye-bot selects the foot-bot that is situated in the center (relative to the hill) and that is closest to the light source. This allows target morphologies to be formed away from the two edges of the hill while allowing completed target morphologies to navigate directly to the light source without colliding with individual foot-bots along the way. Depending on the total number of foot-bots that have acknowledged the issued warning, the eye-bot uses the established communication link to form a chain morphology of either size two or three. That is, if three foot-bots have acknowledged, a chain morphology of size three is formed by sending the signal RED/BLUE to activate the execution of SWARMORPH-script 2 in the

[4]We have empirically determined that images taken at a distance of 30 cm from each other and from a height of 2.42 m (measured from the ground to the tip of the camera) yield the best results in our experimental setting.

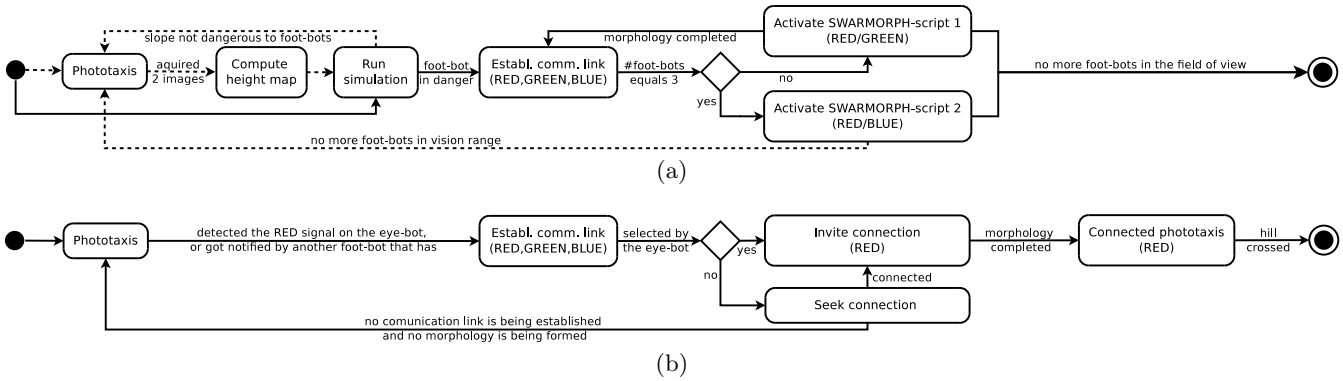


Figure 6: Finite state machine respresentation of the control logic of (a) the eye-bot and (b) the foot-bots. Transitions shown in dashed lines are not considered in this study, as they are only applicable if flight control algorithms are executed on the eye-bot. The signals sent using the LEDs in each state are shown in parentheses.

foot-bots. In all other cases, a chain morphology of size two is formed by sending the signal RED/GREEN to activate the execution of SWARMORPH-script 1. As only one morphology is formed at a time, counting the number of foot-bots sending the RED signal allows the eye-bot to determine when target morphologies are completed. The morphology formation process is iterated until no foot-bots are detected in the field of view. In this manner, by sequentially forming chain morphologies composed of two or three robots, the eye-bot guarantees the crossing of potentially any number of foot-bots given there is more than one foot-bot deployed.

Figure 6b shows the individual states in the behavior-based controller of the foot-bots. Each foot-bot executes a phototaxis behavior to navigate to the light source. The light source is detected using the on-board light sensors. Simultaneously, it uses the upward-pointing camera to detect potential warnings (i.e., the RED signal) issued by the eye-bot. Note that due to the wider field of view of the eye-bot, it is not necessarily the case that a foot-bot in the field of view of the eye-bot has in turn the eye-bot in its field of view. The foot-bots closest to the light source are more probable to detect signals sent by the eye-bot first, as the eye-bot first enters their field of view. Therefore, the foot-bot that detects a RED signal sent by the eye-bot broadcasts a message through the mxRAB device to inform the other foot-bots. All foot-bots become stationary and use the BLUE signal to acknowledge the warning issued by the eye-bot.

Stationary foot-bots use the signals RED, GREEN, and BLUE to establish a one-to-one communication link to the eye-bot. Foot-bots that do not perceive the eye-bot in the field of view or get excluded from the selection process seek for a connection that has to be formed. A connection seeking foot-bot does not move until it is invited by a connection inviting foot-bot. Once the commmunication link is established, the selected foot-bot uses the signal subsequently transmitted by the eye-bot (RED/GREEN or RED/BLUE) to execute the appropriate SWARMORPH-script that leads to the formation of the requested target morphology. The selected foot-bot initiates the morphology formation process by inviting a connection at its rear. Morphologies are completed when connection inviting foot-bots and connection seeking foot-bots execute basic robot behaviors described in a SWARMORPH-script. The underlying connection forming mechanism used in this study allows a connection inviting foot-bot to actively recruit an optimally situated connection seeking foot-bot and guide the recruit to the location where the connection is required. Note that the connection inviting foot-bots and the foot-bots in the selection process use the mxRAB device to send messages that inhibit nearby connection seeking foot-bots from leaving the mophology formation area and driving towards the light. Foot-bots in a completed morphology execute a connected phototaxis behavior that allows the foot-bots to cross the hill as a composite entity. The execution of controllers is stopped on foot-bots that have successfully crossed the hill and reached the light source.

5. EXPERIMENTS AND RESULTS

We performed a series of experiments to assess the performance of the decision-making process (i.e., whether or not to require self-assembly) carried out by the eye-bot. We also validated our approach on real robots using the task and experimental setup described in Sect. 3. The properties of the self-assembly mechanism (i.e., precision, speed and reliability) used by the foot-bots in our approach have already been studied and were presented in [18].

In our experiments, we consider a mock-up of a hill (with a maximum inclination of 30 °) that cannot be crossed by individual foot-bots. Therefore, a successful task completion requires the eye-bot to detect the hazardous situation. We let the eye-bot take 10 different sets of images in which the hill and the 5 foot-bots are always visible. Each set consists of two images taken at 30 cm from each other. The images in each set are used to compute a height map from which a height profile is retrieved for the estimated trajectory of each foot-bot in the field of view. Simulations are run on the resulting 50 height profiles to determine the maximum inclination on each trajectory. The mean of the computed maximum inclination is 29.12 ° with a standard deviation of 4.1 °. This result indicates that, on average, the internal representation of the environment closely correspond to reality. Also, the low standard deviation indicates that the decision-making process is often able to identify the encountered environment as hazardous to individual foot-bots. However, the eye-bot may need to assume a more defensive threshold angle (e.g., 20 °) an individial foot-bot can withstand in order to entirely avoid faulty decisions that can cause foot-bots to topple over.

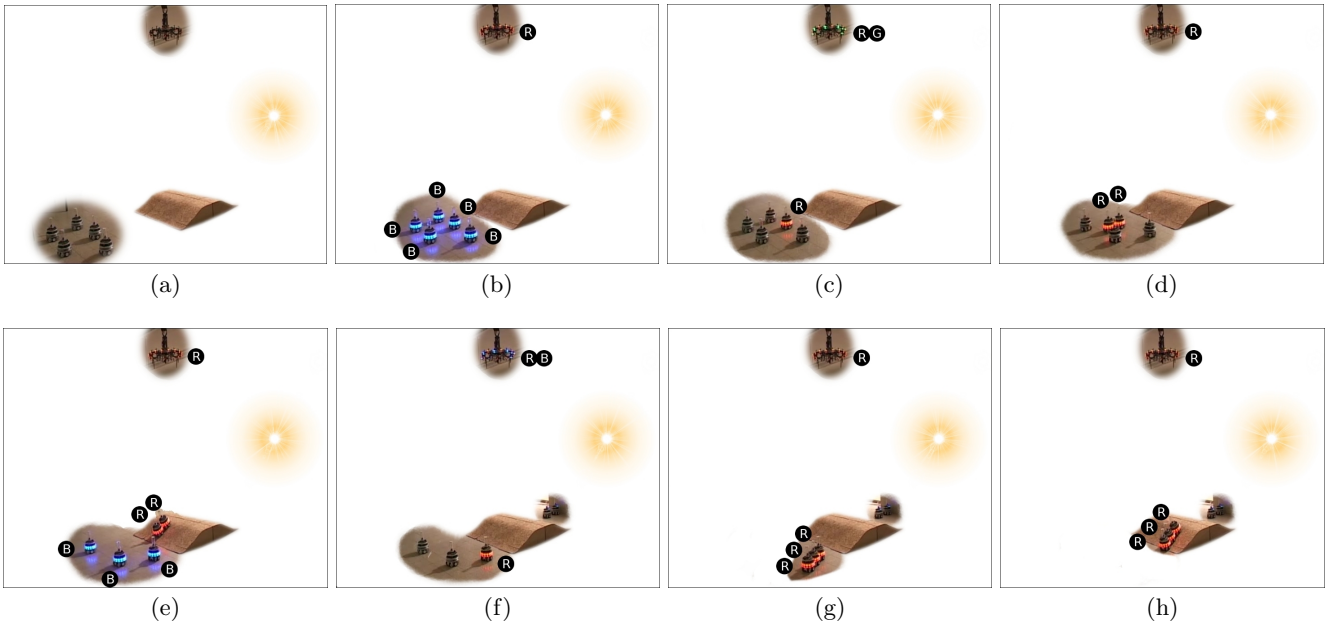


Figure 7: Snapshots of a supervised morphogenesis experiment in the hill-crossing task considered in this paper. For the sake of better visibility, the background has been removed in all images. The letters represent the signals transmited by the robots: R=RED, G=GREEN, B=BLUE, RG=RED/GREEN, and RB=RED/BLUE. (a) Deployment phase. Foot-bots drive towards the light source. The eye-bot runs on-board simulations. (b) The eye-bot attracts the attention of foot-bots using the RED signal: the foot-bots halt. (c) The eye-bot selects a foot-bot and activates the execution of SWARMORPH-script 1 (RED/GREEN signal). (d) A chain morphology composed of two foot-bots is formed. (e) The morphology moves towards the light while the eye-bot selects a next foot-bot and (f) activates the execution of SWARMORPH-script 2 (RED/BLUE signal). (g) A chain morphology composed of three foot-bots is formed. (h) The morphology executes connected phototaxis.

The decision-making process of the eye-bot is, if necessary, followed by the supervised morphology formation process. In our experiments, the stationary eye-bot is not able to acquire the two images required to build the internal representation of the environment. Hence, it uses a pre-calculated height map that was computed using images taken prior to running the experiment. Initially, all five foot-bots execute a phototaxis behavior to navigate to the light source. The eye-bot transmits the RED signal to attract the attention of the foot-bots as soon as on-board simulations detect a slope too steep for individual foot-bots to cross. Immediately before reaching the slope, the closest foot-bots to the light source detect the signal RED on the eye-bot and therefore become stationary. These foot-bots inform their neighboring foot-bots, yet unaware of the hazardous situation, by broadcasting a message through the mxRAB device. In what follows, the eye-bot selects one of the foot-bots able to perceive the eye-bot and establishes a communication link to it. Once the communication link is established, the eye-bot initiates morphology formation by transmitting either the RED/GREEN or the RED/BLUE signal as a function of the total number of foot-bots in the field of view. While completed morphologies move towards the light and successfully cross the hill, the eye-bot continues to supervise the formation of further morphologies using the remaining foot-bots.

Figure 7 shows snapshots of an experimental run of supervised morphogenesis. The complete video footage of the experiment and more details on the results can be found online at `http://iridia.ulb.ac.be/supp/IridiaSupp2011-019/`.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we have enhanced the sensing capabilities of a ground-based self-assembling robotic system by incorporating aerial robots into the system. We introduced supervised morphogenesis – a novel approach that enables aerial robots i) to detect tasks on the ground that cannot be detected by ground-based robots and ii) to initiate and control the formation of morphologies that meet the challenges posed by the task. A key feature of supervised morphogenesis is that the aerial robots perform on-board simulations to evaluate the adequacy of different morphologies to a considered task. We reported on experiments conducted in which supervised morphogenesis was tested on real robotic hardware in a hill-crossing task.

To the best of our knowledge, the work presented in this paper represents the first implementation of a robotic system that enables aerial robots to control morphology formation of ground-based self-assembling robots. We have shown how our approach can be used to enhance the adaptivity of self-assembling robot systems to previously unknown tasks. Our approach does not require any proprietary hardware and only relies on off-the-shelf components such as LEDs and digital cameras to enable communication between the two robot types. Therefore, our approach can also be applied on other robotic platforms.

In our ongoing work, we are studying how to leverage the approach presented in this study to operate in environments composed of various, different tasks that require the formation of a variety of morphologies in parallel. We are also considering to provide aerial robots with the abil-

ity to determine an appropriate morphology to a task by simulating the perceived environment and the robots on the ground using on-board *physics-based* simulations. Moreover, we intend to couple physics-based simulations with machine learning techniques to let aerial robots learn about task-to-morphology mappings. Such a system can feature the ability to produce the most appropriate morphology for potentially any type of physically plausible task and environment.

7. ACKNOWLEDGMENTS

This work was partially supported by the European Commission via the ERC Advance Grant "E-SWARM: Engineering Swarm Intelligence Systems" (grant 246939). Alessandro Stranieri acknowledges support from the MIBISOC network, an Initial Training Network funded by the European Commission, grant PITN–GA–2009–238819. Alexander Scheidler acknowledges support from the postdoc programme of the German Academic Exchange Service (DAAD) and the Meta-X project, funded by the Scientific Research Directorate of the French community of Belgium. Marco Dorigo acknowledges support from the Belgian F.R.S.-FNRS, of which he is a research director.

8. REFERENCES

[1] R. Groß and M. Dorigo. Self-assembly at the macroscopic scale. *Proc. IEEE*, 96(9):1490–1508, 2008.

[2] A. L. Christensen, R. O'Grady, and M. Dorigo. SWARMORPH-script: a language for arbitrary morphology generation in self-assembling robots. *Swarm Intelligence*, 2(2–4):143–165, 2008.

[3] C. Jones and M.J. Matarić. From local to global behavior in intelligent self-assembly. In *Proc. of the 2003 IEEE Int. Conf. on Rob. and Autom.*, pages 721–726. IEEE Computer Society Press, Los Alamitos, CA, 2003.

[4] E. Klavins, R. Ghrist, and D. Lipsky. A grammatical approach to self-organizing robotic systems. *IEEE Trans. Autom. Control*, 51(6):949–962, 2006.

[5] K. Støy and R. Nagpal. Self-reconfiguration using directed growth. In *Proc. of the Int. Conf. on Distr. Auton. Rob. Syst.*, pages 1–10. Springer, Berlin, Germany, 2004.

[6] W. Liu and A. Winfield. Autonomous morphogenesis in self-assembling robots using ir-based sensing and local communications. In *Proc. of the 7th Int. Conf. on Swarm Intelligence*, volume 6234 of *LNCS*, pages 107–118. Springer, Berlin, Germany, 2010.

[7] H. Wei, Y. Chen, M. Liu, Y. Cai, and T. Wang. Swarm robots: From self-assembly to locomotion. *The Computer Journal*, 54(9):1465–1474, 2011.

[8] R. O'Grady, A. L. Christensen, C. Pinciroli, and M. Dorigo. Robots autonomously self-assemble into dedicated morphologies to solve different tasks. In *Proc. of 9th Int. Conf. on Auton. Agents and Multiagent Syst.*, pages 1517–1518. IFAAMAS, Richland, SC, 2010.

[9] R. O'Grady, R. Groß, A. L. Christensen, and M. Dorigo. Self-assembly strategies in a group of autonomous mobile robots. *Autonomous Robots*, 28(4):439–455, 2010.

[10] R. T. Vaughan et al. Fly Spy: Lightweight localization and target tracking for cooperating air and ground robots. In *Proc. of the 5th Int. Symp. on Distrib. Auton. Rob. Syst.*, pages 315–324. Springer, Berlin, Germany, 2000.

[11] A. Stentz, A. Kelly, H. Herman, P. Rander, O. Amidi, and R. Mandelbaum. Integrated air/ground vehicle system for semi-autonomous off-road navigation. In *Proc. of AUVSI Unmanned Syst. Symp.*, 2002.

[12] B. Grocholsky, S. Bayraktar, V. Kumar, C. J. Taylor, and G. Pappas. Synergies in feature localization by air-ground robot teams. In *Proceedings of the 9th International Symposium on Experimental Robotics*, pages 353–362. Springer, Berlin, Germany, 2004.

[13] B. Grocholsky, J. Keller, V. Kumar, and G. Pappas. Cooperative air and ground surveillance. *IEEE Robotics & Automation Magazine*, 13:16–25, 2006.

[14] M. A. Hsieh et al. Adaptive teams of autonomous aerial and ground robots for situational awareness. *Journal of Field Robotics*, 24(11-12):991–1014, 2007.

[15] N. Michael, J. Fink, and V. Kumar. Controlling a team of ground robots via an aerial robot. In *Int. Conf. on Intel. Rob. and Syst.*, pages 965–970. IEEE Press, Piscataway, NJ, 2007.

[16] M. Dorigo et al. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, in press, 2012.

[17] J.F. Roberts, T.S. Stirling, J-C. Zufferey, and D. Floreano. 2.5D infrared range and bearing system for collective robotics. In *Proc. of the 2010 IEEE/RSJ Int. Conf. on Intel. Rob. and Syst.*, pages 3659–3664. IEEE Press, Piscataway, NJ, 2009.

[18] N. Mathews et al. Enhanced directional self-assembly based on active recruitment and guidance. In *Proc. of the 2011 IEEE/RSJ Int. Conf. on Intel. Rob. and Syst.*, pages 4762–4769. IEEE Computer Society Press, Los Alamitos, CA, 2011.

[19] M. Z. Brown, D. Burschka, and G. D. Hager. Advances in computational stereo. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25:993–1008, 2003.

[20] G. Bradski and A. Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly, Cambridge, MA, 2008.

[21] J. Shi and C. Tomasi. Good features to track. In *Proc. of the IEEE Conf. on Comp. Vis. and Pat. Recog.*, pages 593–600. IEEE Press, Piscataway, NJ, 1994.

[22] J.Y. Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker. Description of the algorithm. *Intel Corporation Microprocessor Research Labs*, 2000.

[23] H. Hirschmüller and S. Gehrig. Stereo matching in the presence of sub-pixel calibration errors. In *Proc. of the IEEE Conf. on Comp. Vis. and Pat. Recog.*, pages 437–444. IEEE Press, Piscataway, NJ, 2009.

[24] N. Mathews, A. L. Christensen, E. Ferrante, R. O'Grady, and M. Dorigo. Establishing spatially targeted communication in a heterogeneous robot swarm. In *Proc. of 9th Int. Conf. on Auton. Agents and Multiagent Syst.*, pages 939–946. IFAAMAS, Richland, SC, 2010.

Decentralized Active Robotic Exploration and Mapping for Probabilistic Field Classification in Environmental Sensing

Kian Hsiang Low[†], Jie Chen[†], John M. Dolan[§], Steve Chien[‡], and David R. Thompson[‡]
Department of Computer Science, National University of Singapore, Republic of Singapore[†]
Robotics Institute, Carnegie Mellon University, Pittsburgh PA 15213 USA[§]
Jet Propulsion Laboratory, California Institute of Technology, Pasadena CA 91109 USA[‡]
{lowkh, chenjie}@comp.nus.edu.sg[†], jmd@cs.cmu.edu[§]
{steve.chien, david.r.thompson}@jpl.nasa.gov[‡]

ABSTRACT

A central problem in environmental sensing and monitoring is to classify/label the hotspots in a large-scale environmental field. This paper presents a novel *decentralized active robotic exploration* (DARE) strategy for probabilistic classification/labeling of hotspots in a *Gaussian process* (GP)-based field. In contrast to existing state-of-the-art exploration strategies for learning environmental field maps, the time needed to solve the DARE strategy is independent of the map resolution and the number of robots, thus making it practical for *in situ*, real-time active sampling. Its exploration behavior exhibits an interesting formal trade-off between that of boundary tracking until the hotspot region boundary can be accurately predicted and wide-area coverage to find new boundaries in sparsely sampled areas to be tracked. We provide a theoretical guarantee on the active exploration performance of the DARE strategy: under reasonable conditional independence assumption, we prove that it can optimally achieve two formal cost-minimizing exploration objectives based on the misclassification and entropy criteria. Importantly, this result implies that the uncertainty of labeling the hotspots in a GP-based field is greatest at or close to the hotspot region boundaries. Empirical evaluation on real-world plankton density and temperature field data shows that, subject to limited observations, DARE strategy can achieve more superior classification of hotspots and time efficiency than state-of-the-art active exploration strategies.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Stochastic processes; I.2.9 [**Robotics**]: Autonomous vehicles

General Terms

Algorithms, Performance, Experimentation, Theory

Keywords

Multi-robot exploration and mapping, adaptive sampling, active learning, Gaussian process

1. INTRODUCTION

A fundamental problem in environmental sensing and monitoring is to identify and delineate the hotspot regions in a large-scale environmental field [2, 11]. It involves partitioning the area spanned by the field into one class of regions called the hotspot regions in which the field measurements exceed a predefined threshold, and the other class of regions where they do not. Such a problem arises in many real-world applications such as precision agriculture, monitoring of ocean and freshwater phenomena (e.g., plankton bloom), forest ecosystems, rare species, pollution (e.g., oil spill), or contamination (e.g., radiation leak). In these applications, it is necessary to assess the spatial extent and shape of the hotspot regions accurately due to severe economic, environmental, and health implications, as discussed in [11]. In practice, this aim is non-trivial to achieve because the constraints on the sampling assets' resources (e.g., energy consumption, mission time, sensing range) limit the number and coverage of *in situ* observations over the large field that can be used to infer the hotspot regions. Subject to limited observations, the most informative ones should therefore be selected in order to minimize the uncertainty of estimating the hotspot regions (or, equivalently, classifying/labeling the hotspots) in the large field, which motivates our adaptive sampling work in this paper.

Mobile robot teams are particularly desirable for performing the above environmental sensing task because they can actively explore to map the hotspot regions at high resolution. On the other hand, static sensors lack mobility and are therefore not capable of doing this well unless a large quantity is deployed. While research in multi-robot exploration and mapping have largely focused on the conventional task of building occupancy grids [10], some recent efforts are put into the more complex, general task of sampling spatially distributed environmental fields [4, 5, 6]. In contrast to occupancy grids that assume *discrete, independent* cell occupancies, environmental fields are characterized by *continuous-valued, spatially correlated* measurements, properties of which cannot be exploited by occupancy grid mapping strategies to select the most informative observation paths. To exploit such properties, exploration strategies for learning environmental field maps have recently been developed and can be classified into two regimes: (a) *wide-area coverage* strategies [3, 4, 5] consider sparsely sampled (i.e., largely unexplored) areas to be of high uncertainty and consequently spread observations evenly across the field; (b) *hotspot sampling* strategies [7] assume areas of high uncertainty and interest to contain extreme, highly-varying measurements and hence produce clustered observations. Formal, principled approaches of exploration [4, 5] have also

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

been devised to simultaneously perform hotspot sampling when a hotspot region is found as well as wide-area coverage to search for new hotspot regions in sparsely sampled areas. These strategies optimize their observation paths to minimize the uncertainty (e.g., in terms of mean-squared error or entropy) of mapping the entire continuous-valued field. They are, however, suboptimal for classifying/labeling the hotspots in the field, which we will discuss and demonstrate theoretically and empirically in this paper. More details of their exploration behavior and properties will be provided in Section 6.1.

This paper proposes a novel *decentralized active robotic exploration* (DARE) strategy for probabilistic classification/ labeling of hotspots in a large-scale environmental field (Section 5). The environmental field is assumed to be realized from a rich class of probabilistic spatial models called *Gaussian process* (GP) (Section 2) that can formally characterize its spatial correlation structure. More importantly, it can provide formal measures of classification/labeling uncertainty (i.e., in the form of cost functions) such as the misclassification and entropy criteria (Section 3) for directing the robots to explore highly uncertain areas of the field. The chief impediment to using these formal criteria is that they result in cost-minimizing exploration strategies (Section 4), which cannot be solved in closed form. To resolve this, they are reformulated as reward-maximizing dual strategies, from which we can then derive the approximate DARE strategy to be solved in closed form efficiently. The specific contributions of our work include:

- analyzing the time complexity of solving the DARE strategy (Section 5): we prove that its incurred time is independent of the map resolution and the number of robots, thus making it practical for *in situ*, real-time active sampling. In contrast, existing state-of-the-art exploration strategies [3, 4, 5] for learning environmental field maps scale poorly with increasing map resolution and/or number of robots (Section 6.1);
- analyzing the exploration behavior of the DARE strategy through its formulation (Section 5): it exhibits an interesting formal trade-off between that of boundary tracking until the hotspot region boundary can be accurately predicted and wide-area coverage to find new boundaries in sparsely sampled areas to be tracked. In contrast, ad hoc, reactive boundary tracking strategies [9, 12] typically require a hotspot region boundary to be located manually or via random exploration and are not driven by the need to maximize the fidelity of estimating multiple hotspot regions given limited observations;
- providing theoretical guarantee on the active exploration performance of the DARE strategy (Section 5): we prove that, under reasonable conditional independence assumption, it produces the same optimal observation paths as that of the centralized cost-minimizing strategies, the latter of which otherwise cannot be solved in closed form. This result has a simple but important implication: the uncertainty of labeling the hotspots in a GP-based field is greatest at or close to the hotspot region boundaries;
- empirically evaluating the active exploration performance and time efficiency of the DARE strategy on real-world plankton density and temperature field data (Section 6): subject to limited observations, the DARE strategy can achieve better classification of the hotspots than state-of-the-art active exploration strategies [1, 5] while being significantly more time-efficient than those performing wide-area coverage and hotspot sampling.

2. GAUSSIAN PROCESS-BASED ENVIRONMENTAL FIELD

The *Gaussian process* (GP) can be used to model an environmental field as follows: the environmental field is defined to vary as a realization of a GP. Let $\mathcal{X}$ be a set of sampling locations representing the domain of the environmental field such that each location $x \in \mathcal{X}$ is associated with a realized (random) measurement y_x (Y_x) if x is sampled/observed (unobserved). Let $\{Y_x\}_{x\in\mathcal{X}}$ denote a GP, that is, every finite subset of $\{Y_x\}_{x\in\mathcal{X}}$ has a multivariate Gaussian distribution [8]. The GP is fully specified by its prior mean $\mu_x \triangleq \mathbb{E}[Y_x]$ and covariance $\sigma_{xs} \triangleq \mathrm{cov}[Y_x, Y_s]$ for all $x, s \in \mathcal{X}$. In the experiments (Section 6), we assume that the GP is second-order stationary, i.e., it has a constant *prior* mean and a stationary *prior* covariance structure (i.e., σ_{xs} is a function of $x - s$ for all $x, s \in \mathcal{X}$). The prior mean and covariance structure of the GP are assumed to be known. Let $\mathcal{S}$ denote a subset of locations of $\mathcal{X}$ sampled a priori (either by the robot team or other sampling assets) and $y_\mathcal{S}$ be a row vector of corresponding measurements. Given the set $\mathcal{S}$ of sampled locations and corresponding measurements $y_\mathcal{S}$, the distribution of Y_x at any unobserved location $x \in \mathcal{X} \setminus \mathcal{S}$ remains Gaussian with the following *posterior* mean and variance

$$\mu_{x|\mathcal{S}} = \mu_x + \Sigma_{x\mathcal{S}}\Sigma_{\mathcal{S}\mathcal{S}}^{-1}(y_\mathcal{S} - \mu_\mathcal{S})^\top \quad (1)$$

$$\sigma_{x|\mathcal{S}}^2 = \sigma_x^2 - \Sigma_{x\mathcal{S}}\Sigma_{\mathcal{S}\mathcal{S}}^{-1}\Sigma_{\mathcal{S}x} \quad (2)$$

where $\mu_\mathcal{S}$ is a row vector with mean components μ_s for every location $s \in \mathcal{S}$, $\Sigma_{x\mathcal{S}}$ is a row vector with covariance components σ_{xs} for every location $s \in \mathcal{S}$, $\Sigma_{\mathcal{S}x}$ is the transpose of $\Sigma_{x\mathcal{S}}$, and $\Sigma_{\mathcal{S}\mathcal{S}}$ is a covariance matrix with components $\sigma_{ss'}$ for every pair of locations $s, s' \in \mathcal{S}$. To map the entire field, the measurements at its unobserved areas can be predicted using the posterior mean (1) and the uncertainty of each of these point-based predictions is represented by the posterior variance (2). An important property of GP is that the posterior variance $\sigma_{x|\mathcal{S}}^2$ (2) is independent of the observed measurements $y_\mathcal{S}$.

If the environmental field evolves over time, then its domain is extended to include the temporal dimension: let $\mathcal{X}$ instead denote a set of *spatiotemporal* inputs such that each input $x \in \mathcal{X}$ comprises both the spatial location and time. The rest of the GP model formulation remains unchanged.

3. COST FUNCTIONS

Recall that the exploration objective is to select observation paths that minimize the uncertainty of estimating the hotspot regions in the field. To achieve this, formal measures of uncertainty (specifically, in the form of cost functions) have to be defined. Let us first consider the feasibility of using cost functions that quantify the uncertainty of mapping the entire continuous-valued field, such as (a) sum of posterior variances (2) over the unobserved locations in $\mathcal{X} \setminus \mathcal{S}$ [4]

$$\sum_{x\in\mathcal{X}\setminus\mathcal{S}} \sigma_{x|\mathcal{S}}^2 \quad (3)$$

and (b) posterior joint entropy of the measurements $Y_{\mathcal{X}\setminus\mathcal{S}}$ at the unobserved locations in $\mathcal{X} \setminus \mathcal{S}$ [5]

$$\mathbb{H}[Y_{\mathcal{X}\setminus\mathcal{S}}|y_\mathcal{S}] \triangleq -\int P(y_{\mathcal{X}\setminus\mathcal{S}}|y_\mathcal{S}) \log P(y_{\mathcal{X}\setminus\mathcal{S}}|y_\mathcal{S}) \, \mathrm{d}y_{\mathcal{X}\setminus\mathcal{S}} \ .$$

These cost functions have been utilized in [4, 5] to guide exploration: the resulting active exploration strategies for learning GP-based field maps are non-adaptive and perform wide-area coverage, that is, observation paths are distributed evenly across the field. Do these wide-area coverage strategies also optimize our exploration objective or should observation paths be directed to sample specific features of the field instead? In the rest of this paper, we will show that, by defining cost functions to measure the uncertainty of classifying the hotspots in the field, our objective can be better achieved by performing the latter.

Let us begin by framing the problem of estimating the hotspot regions in a field formally as one of classifying/labeling the hotspots in the field: A location x is defined as a hotspot if its corresponding field measurement Y_x is greater than or equal to a predefined threshold, denoted by γ. Let $\{Z_x\}_{x\in\mathcal{X}}$ denote a binary random process such that Z_x is an indicator variable of label 1 if $Y_x \geq \gamma$ (i.e., location x is a hotspot), and label 0 otherwise. Then, our problem of estimating the hotspot regions is equivalent to one of labeling the hotspots in the field, specifically, by predicting the label of Z_x for every location $x \in \mathcal{X}$. As a result, our exploration objective can be achieved through the use of cost functions that measure the uncertainty of labeling the hotspots in the field. Two such cost functions will be defined next.

Let $\widehat{Z}_x$ be the predicted label of Z_x for every location $x \in \mathcal{X}$ and the cost of predicting (or, more precisely, misclassifying) the label of Z_x with $\widehat{Z}_x$ be denoted by the following $0-1$ loss function

$$L(Z_x, \widehat{Z}_x) = \left|Z_x - \widehat{Z}_x\right| = \begin{cases} 1 & \text{if } Z_x \neq \widehat{Z}_x \ , \\ 0 & \text{otherwise.} \end{cases} \tag{4}$$

That is, (4) counts a false positive (i.e., the location x is labeled as a hotspot but it is not) or false negative (i.e., x is not labeled as a hotspot but it is) as a misclassification. If Z_x is unlabeled (i.e., location x is unobserved), then we calculate the *expected* cost (or risk) of predicting the label of Z_x with $\widehat{Z}_x$ instead, which is denoted by

$$\begin{aligned} R_{\widehat{Z}_x|\mathcal{S}} = & \ \textstyle\sum_{i=0}^{1} L(Z_x = i, \widehat{Z}_x)\ P(Z_x = i|y_\mathcal{S}) \\ = & \ \widehat{Z}_x\ (1 - P(Z_x = 1|y_\mathcal{S})) + (1 - \widehat{Z}_x)\ P(Z_x = 1|y_\mathcal{S}) \\ = & \ P(\widehat{Z}_x \neq Z_x|y_\mathcal{S}) \end{aligned} \tag{5}$$

where $P(Z_x = 1|y_\mathcal{S}) = P(Y_x \geq \gamma|y_\mathcal{S})$, the second equality results from $P(Z_x = 0|y_\mathcal{S}) = 1 - P(Z_x = 1|y_\mathcal{S})$, and the last equality states that the risk (5) is equal to the probability of misclassification.

The risk (5) is minimized by the Bayes decision/classification rule

$$\begin{aligned} \widehat{Z}^*_x = & \begin{cases} 1 & \text{if } P(Z_x = 1|y_\mathcal{S}) \geq 0.5 \ , \\ 0 & \text{otherwise.} \end{cases} \\ = & \ \arg\max_{i\in\{0,1\}} P(Z_x = i|y_\mathcal{S}) \ . \end{aligned}$$

Using $\widehat{Z}^*_x$ as the predicted label of Z_x, the risk (5) reduces to

$$R_{\widehat{Z}^*_x|\mathcal{S}} = \min\left(P(Z_x = 1|y_\mathcal{S}), 1 - P(Z_x = 1|y_\mathcal{S})\right) \ . \tag{6}$$

Consequently, the sum of risks (or expected number of misclassifications) over the unobserved locations in $\mathcal{X} \setminus \mathcal{S}$ is

$$\sum_{x\in\mathcal{X}\setminus\mathcal{S}} R_{\widehat{Z}^*_x|\mathcal{S}} \ , \tag{7}$$

which defines our first cost function. We call this (7) the *misclassification criterion*.

The second cost function, which we call the *entropy criterion*, is defined as the posterior joint entropy of the labels of $Z_{\mathcal{X}\setminus\mathcal{S}}$ at the unobserved locations in $\mathcal{X} \setminus \mathcal{S}$

$$\mathbb{H}[Z_{\mathcal{X}\setminus\mathcal{S}}|y_\mathcal{S}] \ . \tag{8}$$

4. CENTRALIZED ACTIVE EXPLORATION

In this section, we will formulate greedy cost-minimizing exploration strategies based on the misclassification (7) and entropy (8) criteria defined in Section 3. Unfortunately, these centralized strategies cannot be evaluated in closed form, as explained in this section. To resolve this, these cost-minimizing strategies must first be reformulated as reward-maximizing dual strategies, from which we can then derive the approximate DARE strategy (Section 5) to be solved in closed form efficiently.

Supposing the misclassification criterion (7) is used and a set $\mathcal{S}$ of locations are previously sampled, the exploration strategy for directing a team of k robots has to select the next set $\mathcal{O} \subseteq \mathcal{X} \setminus \mathcal{S}$ of k locations to be observed that minimize the sum of *expected* risks:

$$\min_{\mathcal{O}} \sum_{x\in\mathcal{X}\setminus\mathcal{S}} \mathbb{E}_{Y_\mathcal{O}|y_\mathcal{S}}\left\{R_{\widehat{Z}^*_x|\mathcal{S}\cup\mathcal{O}}\right\} \ . \tag{9}$$

This cost-minimizing strategy (9) can be reformulated as the following reward-maximizing dual strategy, which selects the next set $\mathcal{O}$ of locations to be observed that maximize the sum of expected risk reductions:

$$\max_{\mathcal{O}} \sum_{x\in\mathcal{X}\setminus\mathcal{S}} R_{\widehat{Z}^*_x|\mathcal{S}} - \mathbb{E}_{Y_\mathcal{O}|y_\mathcal{S}}\left\{R_{\widehat{Z}^*_x|\mathcal{S}\cup\mathcal{O}}\right\} \ . \tag{10}$$

The equivalence between these two strategies follows immediately from observing that the first term $\sum_{x\in\mathcal{X}\setminus\mathcal{S}} R_{\widehat{Z}^*_x|\mathcal{S}}$ in (10) remains constant with any choice of $\mathcal{O}$. Both strategies cannot be solved exactly due to the expectation term, which cannot be evaluated in closed form.

If the entropy criterion (8) is used instead, then the exploration strategy has to select the next set $\mathcal{O}$ of locations to be observed that minimize the *expected* posterior joint entropy of the labels of $Z_{\mathcal{X}\setminus(\mathcal{S}\cup\mathcal{O})}$:

$$\min_{\mathcal{O}}\ \mathbb{E}_{Z_\mathcal{O}|y_\mathcal{S}}\left\{\mathbb{H}[Z_{\mathcal{X}\setminus(\mathcal{S}\cup\mathcal{O})}|y_\mathcal{S}, Z_\mathcal{O}]\right\} \ . \tag{11}$$

This cost-minimizing strategy (11) can be reformulated as the following reward-maximizing dual strategy, which selects the next set $\mathcal{O}$ of locations with maximum label entropy to be observed:

$$\max_{\mathcal{O}}\ \mathbb{H}[Z_\mathcal{O}|y_\mathcal{S}] \ . \tag{12}$$

To show their equivalence, $\mathbb{H}[Z_{\mathcal{X}\setminus\mathcal{S}}|y_\mathcal{S}]$ (8) is first expanded using chain rule of entropy:

$$\mathbb{H}[Z_{\mathcal{X}\setminus\mathcal{S}}|y_\mathcal{S}] = \mathbb{H}[Z_\mathcal{O}|y_\mathcal{S}] + \mathbb{E}_{Z_\mathcal{O}|y_\mathcal{S}}\left\{\mathbb{H}[Z_{\mathcal{X}\setminus(\mathcal{S}\cup\mathcal{O})}|y_\mathcal{S}, Z_\mathcal{O}]\right\} \ . \tag{13}$$

From (13), since $\mathbb{H}[Z_{\mathcal{X}\setminus\mathcal{S}}|y_\mathcal{S}]$ is a constant, the choice of $\mathcal{O}$ that maximizes $\mathbb{H}[Z_\mathcal{O}|y_\mathcal{S}]$ (i.e., (12)) will also minimize $\mathbb{E}_{Z_\mathcal{O}|y_\mathcal{S}}\left\{\mathbb{H}[Z_{\mathcal{X}\setminus(\mathcal{S}\cup\mathcal{O})}|y_\mathcal{S}, Z_\mathcal{O}]\right\}$ (i.e., (11)). When $|\mathcal{O}| = k \geq 2$, both strategies cannot be solved exactly due to the entropy terms, which contain multivariate Gaussian cumulative distribution functions that cannot be evaluated in closed form.

5. DECENTRALIZED ACTIVE EXPLORATION

This section presents a novel *decentralized active robotic exploration* (DARE) strategy that can approximately achieve both cost-minimizing exploration objectives (9) and (11) (Section 4) based on the misclassification and entropy criteria, respectively. Unlike the centralized cost-minimizing and reward-maximizing exploration strategies (Section 4), the DARE strategy can be solved in closed form efficiently.

The DARE strategy for directing each of the k robots has to select the next location $x \in \mathcal{X} \setminus \mathcal{S}$ to be observed that trades off between (a) minimizing the difference between its predicted measurement $\mu_{x|\mathcal{S}}$ and the boundary threshold γ, and (b) maximizing the square root of its posterior variance $\sigma^2_{x|\mathcal{S}}$:

$$\min_x \ |\gamma - \mu_{x|\mathcal{S}}|/\sigma_{x|\mathcal{S}} \ . \tag{14}$$

Intuitively, the behavior of the DARE strategy exhibits an interesting trade-off between that of (a) boundary tracking and (b) wide-area coverage: it simultaneously tracks a hotspot region boundary that is found until it can be accurately predicted as well as searches for new hotspot region boundaries in sparsely sampled areas to be tracked.

In this paper, the domain $\mathcal{X}$ of the field is assumed to be a grid of sampling locations. The next location x to be observed by each robot is then constrained to be selected from the 4-connected neighborhood $\mathcal{N}$ of the robot's current location instead of from $\mathcal{X} \setminus \mathcal{S}$.

THEOREM 1 (TIME COMPLEXITY). *Solving the DARE strategy (14) requires* $\mathcal{O}(|\mathcal{S}|^2(|\mathcal{S}| + |\mathcal{N}|))$ *time.*

The above result reveals that the time needed to compute the DARE strategy is independent of the map resolution (i.e., domain size $|\mathcal{X}|$) and the number k of robots, thus making it practical for *in situ*, real-time active sampling.

THEOREM 2 (COMMUNICATION OVERHEAD). *Let the communication overhead be the number of broadcast messages sent by each robot over the network. Then, the asynchronous communication overhead of DARE strategy (14) is* $\mathcal{O}(1)$.

In terms of data sharing, each robot broadcasts a message to the other robots sharing its sampled observations since its last broadcast. Coordination between robots is needed only if their neighborhoods intersect: in this case, they may select the same next location to be observed. To avoid this, each robot can broadcast on the same or another message sharing its selected location to be observed next. With asynchronous (e.g., turn-based) communication, the remaining robots avoid choosing prior selected locations. With synchronous communication, for each location in conflict, the higher-numbered robot (obtained by numbering robots uniquely) uses (14) to choose new unselected location to be observed. This is iterated until the conflicts are resolved. This process is not communication-expensive as every location is in conflict with at most 4 robots in its neighborhood.

Under conditional independence assumption, the DARE strategy (14) produces the same observation paths as that of the centralized cost-minimizing strategies (9) and (11) (Section 4), as established in the result below:

THEOREM 3 (PERFORMANCE GUARANTEE). *If the unobserved measurements* $Y_{\mathcal{X}\setminus\mathcal{S}}$ *are conditionally independent given the sampled measurements* $y_{\mathcal{S}}$*, then the DARE strategy (14) is equivalent to both cost-minimizing strategies (9) and (11) based on the misclassification and entropy criteria.*

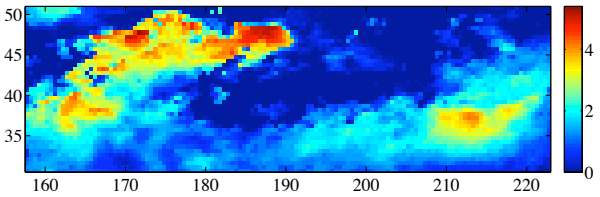


Figure 1: Temperature field bounded within lat. $30.75 - 50.75$**N and lon.** $157.75 - 222.25$**E:** γ **is set to** $3\,^\circ$C**, which results in a hotspot region in the top left and another one in the bottom right.**

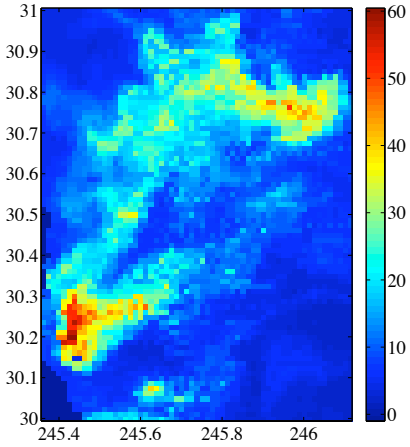


Figure 2: Plankton density field bounded within lat. $30 - 31$**N and lon.** $245.3625 - 246.1125$**E:** γ **is set to** 30 **mg/m**^{3}**, which results in a hotspot region in the top right and another one in the bottom left.**

The proof of the above result can be found in the appendix. The proof construction in fact describes how the DARE strategy (14) can be derived from either reward-maximizing dual strategy (i.e., (10) or (12)) (Section 4). A simple but important implication of this result is that the uncertainty of estimating the hotspot regions in a GP-based field (i.e., in terms of misclassification or entropy criterion) is greatest at or close to the hotspot region boundaries.

In practice, how reasonable is the conditional independence assumption? Firstly, such an assumption is often made in order to calculate the widely-used sum of posterior variances (i.e., mean-squared error) criterion (3) [2, 4]. Secondly, we conjecture that the assumption becomes less restrictive (i.e., Theorem 3 becomes more reliable) when the number $|\mathcal{S}|$ of sampled locations increases to potentially reduce the degree of violation of conditional independence, the spatial correlation between field measurements decreases, and the robots are sufficiently far apart (this last case applies only to the entropy criterion).

6. EXPERIMENTS AND DISCUSSION

This section evaluates the active exploration performance of the DARE strategy (14) empirically on 2 real-world spatial datasets off the west coast of USA: (a) August 2009 AVHRR temperature data (Fig. 1), and (b) March 2009 MODIS plankton density data (Fig. 2). These regions are discretized, respectively, into (a) 130×41 (i.e., $|\mathcal{X}| = 5330$) and (b) 61×81 (i.e., $|\mathcal{X}| = 4941$) grids of sampling locations. Each location x is, respectively, associated with (a) temperature measurement y_x in $^\circ$C, and (b) chlorophyll-a

Table 1: Comparison of active exploration strategies (WC: Wide-area Coverage, HS: Hotspot Sampling, BT: Boundary Tracking).

Exploration strategy	Behavior	Coordination type	Time complexity	Map resolution $\lvert\mathcal{X}\rvert$	No. of robots k
Maximize mutual information [3]	WC	Centralized	$\mathcal{O}(\lvert\mathcal{N}\rvert^k\lvert\mathcal{X}\rvert^2(\lvert\mathcal{X}\rvert+k^2))$	Cubic	Exponential
Minimize sum of variances [4]	WC	Centralized	$\mathcal{O}(\lvert\mathcal{N}\rvert^k\lvert\mathcal{S}\rvert^2\lvert\mathcal{X}\rvert)$	Linear	Exponential
MES [5]	WC	Centralized	$\mathcal{O}(\lvert\mathcal{N}\rvert^k\lvert\mathcal{S}\rvert^2(\lvert\mathcal{S}\rvert+k^2))$	Independent	Exponential
MES+HS [5]	WC+HS	Centralized	$\mathcal{O}(\lvert\mathcal{N}\rvert^k\lvert\mathcal{S}\rvert^2(\lvert\mathcal{S}\rvert+k^2))$	Independent	Exponential
Straddle [1]	WC+BT	Decentralized[1]	$\mathcal{O}(\lvert\mathcal{S}\rvert^2(\lvert\mathcal{S}\rvert+\lvert\mathcal{N}\rvert))$	Independent	Independent
DARE	WC+BT	Decentralized	$\mathcal{O}(\lvert\mathcal{S}\rvert^2(\lvert\mathcal{S}\rvert+\lvert\mathcal{N}\rvert))$	Independent	Independent

(chl-a) measurement y_x in mg/m^3. Using a team of $k = 2, 4, 8$ robots, each robot is tasked to, respectively, explore $1250, 625, 312$ locations in its path to sample a total of about 2500 observations. The simulated robot team is given 120 randomly selected observations as prior data before exploration. We use 2000 randomly selected observations to learn the hyperparameters (i.e., mean and covariance structure) of GP through maximum likelihood estimation [8].

6.1 Comparing Active Exploration Strategies

Since the domains $\mathcal{X}$ of both fields are considerably large, it is prohibitively expensive to compare meaningfully with the wide-area coverage strategies [3, 4] that scale poorly with increasing map resolution and are thus not practical for *in situ*, real-time active sampling. For example, it was reported in [6] that the greedy mutual information-based strategy of [3] incurred more than 62 hours to generate paths for 3 robots to sample a total of 267 observations in a grid of only $\lvert\mathcal{X}\rvert = 1424$ locations. The performance of the DARE strategy is therefore compared to that of three state-of-the-art exploration strategies whose incurred times are independent of the map resolution: (a) The decentralized[1] *straddle* strategy [1] for directing each robot selects the next location x to be observed using $\max_x 1.96\sigma_{x|\mathcal{S}} - \lvert\gamma - \mu_{x|\mathcal{S}}\rvert$. Similar to DARE, its exploration behavior is a trade-off between that of boundary tracking and wide-area coverage. Unlike DARE, its trade-off has to be manually adjusted using an arbitrary weight that, if set inappropriately, may produce suboptimal behavior. For example, this weight is proposed by [1] to be set to 1.96, which is empirically demonstrated later to emphasize boundary tracking more than wide-area coverage. As a result, it tends to persist in tracking boundaries that are already well-predicted before deciding to search for new ones. Subject to limited observations, it may consequently not perform as well as DARE in a field with multiple hotspot regions. Also, it is not known how or whether the value of this weight can be formally derived in order for the straddle strategy to achieve the cost-minimizing exploration objectives (9) and (11); (b) The centralized *maximum entropy sampling* (MES) strategy [5] for directing the robot team performs only wide-area coverage by selecting the next set $\mathcal{O}$ of locations with maximum entropy to be observed using $\max_{\mathcal{O}} \mathbb{H}[Y_{\mathcal{O}}|y_{\mathcal{S}}]$; (c) It can be coupled with hotspot sampling (HS) by modifying the exploration objective to $\max_{\mathcal{O}} \mathbb{H}[Y_{\mathcal{O}}|y_{\mathcal{S}}] + \sum_{x\in\mathcal{O}} \mu_{x|\mathcal{S}}$. We call this the MES+HS strategy [5]. For these centralized strategies, the joint action space is exponential in the number of robots. So, they scale poorly with increasing number of robots. Table 1 summarizes and compares the characteristics of the above-mentioned active exploration strategies; it does not include the communication overhead, which is $\mathcal{O}(1)$ for all strategies.

[1]The original straddle strategy proposed by [1] is developed for a single robot. To transform it into a decentralized multi-robot strategy, we simply execute the single-robot straddle strategy on every robot in the team.

6.2 Performance Metric

The first performance metric used to evaluate the tested strategies is the number of misclassifications

$$M(\mathcal{A}) \triangleq \sum_{x\in\mathcal{A}} L(z_x, \widehat{Z}^*_x)$$

over all locations in a given set $\mathcal{A}$ where the function L is previously defined in (4). Three cases are considered:

(a) $\mathcal{A} = \mathcal{X}$ (i.e., all locations in the domain of the field),

(b) $\mathcal{A} = \mathcal{X}'$ where

$$\mathcal{X}' = \{x \in \mathcal{X} \mid \lvert\gamma - y_x\rvert \le 0.2(\max_{x'\in\mathcal{X}} y_{x'} - \min_{x'\in\mathcal{X}} y_{x'})\}$$

(i.e., all locations with measurements that are close to the boundary threshold of 30 mg/m^3 for the plankton density field and 3 °C for the temperature field), and

(c) $\mathcal{A} = \mathcal{X} \setminus \mathcal{X}'$.

We observe that $\lvert\mathcal{X}'\rvert$ is only about 22% of $\lvert\mathcal{X}\rvert$ for both fields. The second metric is the time taken to compute a strategy.

6.3 Temperature Field Data

Fig. 3 shows the results of the performance of tested strategies averaged over 5 randomly generated starting robot locations for the temperature field. In terms of the $M(\mathcal{X})$ performance, Figs. 3a−3c show that the DARE strategy quickly outperforms the MES and MES+HS strategies as the number of observations increases: their performance differences have been verified using t-tests ($\alpha = 0.1$) to be statistically significant after a total of 500, 750, and 800 observations sampled by teams of 2, 4, and 8 robots, respectively. Hence, the boundary-tracking DARE strategy reduces a greater number of misclassifications over the entire field than wide-area coverage and hotspot sampling. With more observations, the DARE strategy can also perform better than the straddle strategy: their performance differences have been verified using t-tests ($\alpha = 0.1$) to be statistically significant after a total of 500, 1000, and 1600 observations sampled by teams of 2, 4, and 8 robots, respectively. To explain this, we examine the observation paths of a team of 2 robots in one of the 5 test runs, as shown in Fig. 4. The initial performance of the DARE and straddle strategies are similar because they are both searching for hotspot region boundaries (Fig. 4a). As the number of observations increases further, DARE's performance improves over that of the straddle strategy because we observe that it directs the robots to search for new boundaries when the ones that are currently being tracked are well-predicted. In contrast, the

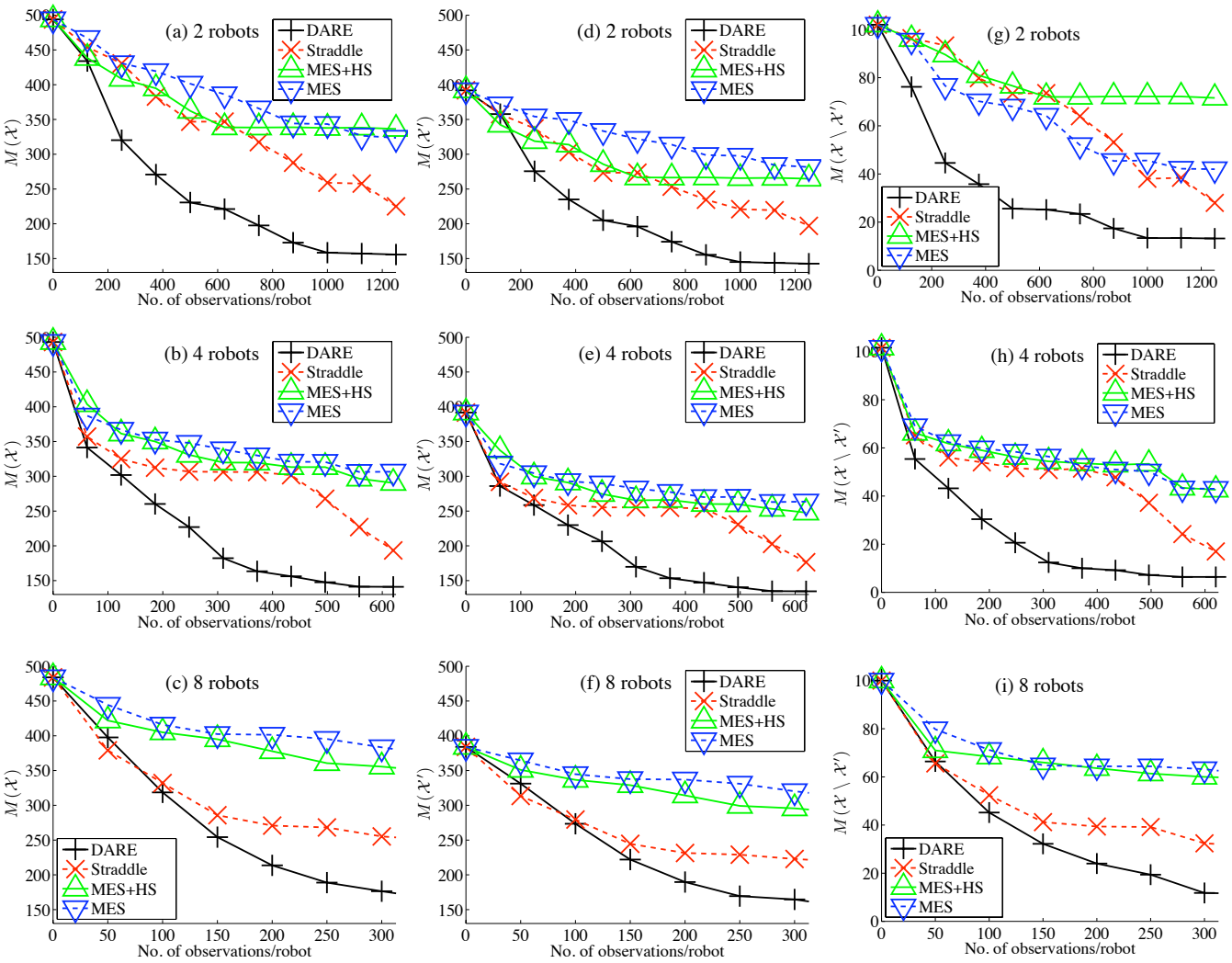


Figure 3: Graphs of (a-c) $M(\mathcal{X})$, (d-f) $M(\mathcal{X}')$, and (g-i) $M(\mathcal{X} \setminus \mathcal{X}')$ vs. no. of observations/robot for varying number of robots actively exploring the temperature field.

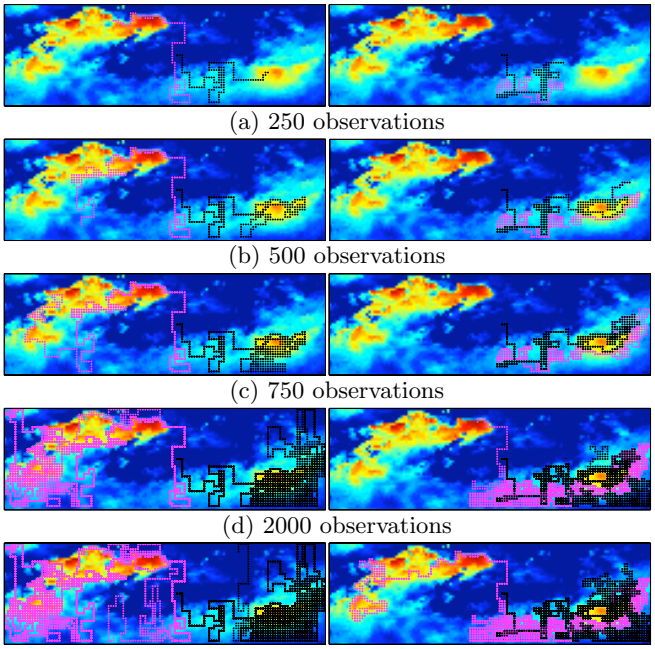
straddle strategy tends to persist in tracking boundaries that are already well-predicted before deciding to search for new ones (Figs. 4b−4e). In terms of the $M(\mathcal{X}')$ and $M(\mathcal{X} \setminus \mathcal{X}')$ performance, Figs. 3d−3i reveal that, with increasing observations, the DARE strategy also reduces a greater number of misclassifications than the other evaluated strategies whether they are over locations close to the boundaries (i.e., in $\mathcal{X}'$) or away from the boundaries (i.e., in $\mathcal{X} \setminus \mathcal{X}'$). It is interesting to note that locations close to the boundaries incur the majority of the misclassifications as compared to those away from the boundaries, which further corroborates the implication of Theorem 3 that there is higher uncertainty in labeling the locations close to the hotspot region boundaries.

6.4 Plankton Density Field Data

Fig. 5 shows the results of the performance of tested strategies averaged over 5 randomly generated starting robot locations for the plankton density field. The results are very similar to that of the temperature field (Section 6.3) except that the performance of the straddle strategy approaches that of the DARE strategy with excessive observations: their performance differences have been verified using t-tests ($\alpha = 0.1$) not to be statistically significant after a total of 2000 and 2240 observations sampled by teams of 2 and 4 robots, respectively. This is expected because the straddle strategy can track and predict the boundaries as well as the DARE strategy given a long enough exploration. However, subject to limited observations (which is more practical, as explained in Section 1), the performance of the DARE strategy is clearly superior to that of the straddle strategy.

6.5 Incurred Time

Fig. 6 shows the results of the time taken to compute the tested strategies averaged over 5 randomly generated starting robot locations for the temperature field; the results of incurred time for the plankton density field are very similar and therefore not shown here. The time differences between the DARE and straddle strategies have been verified using t-tests ($\alpha = 0.1$) not to be statistically significant, which is expected due to the same time complexities, as shown in Table 1. The time taken to compute these two decentralized boundary tracking strategies are shorter than that needed to compute the centralized wide-area coverage and



(a) 250 observations

(b) 500 observations

(c) 750 observations

(d) 2000 observations

(e) 2500 observations

Figure 4: Evolution of 2-robot observation paths produced by DARE (left column) and straddle (right column) strategies sampling a total of (a) 250, (b) 500, (c) 750, (d) 2000, and (e) 2500 observations. The robots start at locations of different lat. 37.75N and 40.75N and same lon. 192.75E.

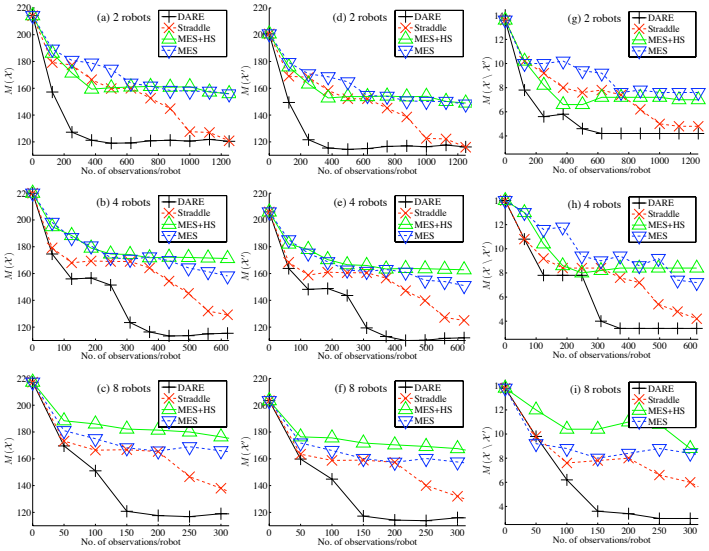


Figure 5: Graphs of (a-c) $M(\mathcal{X})$, (d-f) $M(\mathcal{X}')$, and (g-i) $M(\mathcal{X}\backslash\mathcal{X}')$ vs. no. of observations/robot for varying number of robots actively exploring the plankton density field.

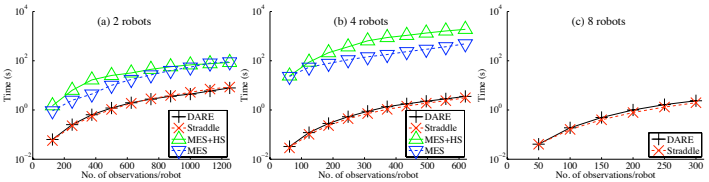


Figure 6: Graphs of incurred time vs. no. of observations/robot for varying number of robots actively exploring the temperature field.

hotspot sampling strategies (i.e., MES+HS and MES) by about one and two orders of magnitude for the cases of 2 and 4 robots, respectively. With a larger number of robots, it can be observed from Fig. 6 that the centralized wide-area coverage and hotspot sampling strategies incur significantly more time because their time complexities are exponential in the number of robots, as shown in Table 1. In contrast, the time incurred by the decentralized boundary tracking strategies do not increase because their time complexities are independent of the number of robots. Note that Fig. 6 does not show the graphs of time taken to compute the centralized strategies for the case of 8 robots because they incur significantly more time than that of 4 robots and their incurred time consequently cannot be recorded correctly due to long integer overflow in C's clock function.

7. CONCLUSION

This paper describes a decentralized active robotic exploration strategy for probabilistic classification of hotspots in a large-scale GP-based environmental field. It has the practical advantage of being significantly more time-efficient over existing state-of-the-art active exploration strategies [3, 4, 5] because its incurred time is independent of the map resolution and the number of robots. In terms of active exploration performance, we have theoretically guaranteed that, under reasonable conditional independence assumption, the DARE strategy can optimally achieve the formal cost-minimizing exploration objectives based on the misclassification and entropy criteria, both of which otherwise cannot be optimized exactly to yield closed-form solutions. We have demonstrated theoretically and empirically that the uncertainty of labeling the hotspots in a GP-based field is greatest at or close to the hotspot region boundaries. The DARE strategy is capable of exploiting this to produce an exploration behavior that formally trades off between that of boundary tracking until the hotspot region boundary can be accurately predicted and wide-area coverage to find new boundaries in sparsely sampled areas to be tracked. Empirical evaluation on real-world plankton density and temperature field data shows that, given limited observations, the DARE strategy can reduce a greater number of misclassifications than state-of-the-art active exploration strategies.

8. ACKNOWLEDGMENTS

This work was supported by MOE AcRF Tier 1 R-252-000-426-133. A portion of this work was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

9. REFERENCES

[1] B. Bryan, J. G. Schneider, R. Nichol, C. Miller, C. Genovese, and L. A. Wasserman. Active learning for identifying function threshold boundaries. In Y. Weiss, B. Schölkopf, and J. Platt, editors, *Advances in Neural Information Processing Systems 18*, pages 163–170, Cambridge, MA, 2006. MIT Press.

[2] N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, NY, 2nd edition, 1993.

[3] A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. *JMLR*, 9:235–284, 2008.

[4] K. H. Low, J. M. Dolan, and P. Khosla. Adaptive multi-robot wide-area exploration and mapping. In *Proc. AAMAS*, pages 23–30, 2008.

[5] K. H. Low, J. M. Dolan, and P. Khosla. Information-theoretic approach to efficient adaptive path planning for mobile robotic environmental sensing. In *Proc. ICAPS*, pages 233–240, 2009.

[6] K. H. Low, J. M. Dolan, and P. Khosla. Active Markov information-theoretic path planning for robotic environmental sensing. In *Proc. AAMAS*, pages 753–760, 2011.

[7] K. H. Low, G. J. Gordon, J. M. Dolan, and P. Khosla. Adaptive sampling for multi-robot wide-area exploration. In *Proc. IEEE ICRA*, pages 755–760, 2007.

[8] C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. MIT Press, Cambridge, MA, 2006.

[9] R. N. Smith, A. Pereira, Y. Chao, P. P. Li, D. A. Caron, B. H. Jones, and G. S. Sukhatme. Autonomous underwater vehicle trajectory design coupled with predictive ocean models: A case study. In *Proc. IEEE ICRA*, pages 4770–4777, 2010.

[10] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, Cambridge, MA, 2005.

[11] R. Webster and M. Oliver. *Geostatistics for Environmental Scientists*. John Wiley & Sons, Inc., NY, 2nd edition, 2007.

[12] F. Zhang and N. E. Leonard. Cooperative control and filtering for cooperative exploration. *IEEE Trans. Automat. Contr.*, 55(3):650–663, 2010.

APPENDIX

Proof of Theorem 3

In Section 4, we have already shown the equivalence between the cost-minimizing and reward-maximizing strategies based on the misclassification and entropy criteria. Therefore, it suffices to prove that the DARE strategy (14) is equivalent to the reward-maximizing strategies.

Let us first prove that the reward-maximizing strategy (10) for the misclassification criterion is equivalent to the DARE strategy (14). From (10),

$$\begin{aligned}
&\max_{\mathcal{O}} \sum_{x\in\mathcal{X}\setminus\mathcal{S}} R_{\widehat{Z}^*_x|\mathcal{S}} - \mathbb{E}_{Y_\mathcal{O}|y_\mathcal{S}}\Big\{R_{\widehat{Z}^*_x|\mathcal{S}\cup\mathcal{O}}\Big\}\\
&= \max_{\mathcal{O}} \sum_{x\in\mathcal{O}} R_{\widehat{Z}^*_x|\mathcal{S}} + \sum_{x\in\mathcal{X}\setminus(\mathcal{S}\cup\mathcal{O})}\Big(R_{\widehat{Z}^*_x|\mathcal{S}} - \mathbb{E}_{Y_\mathcal{O}|y_\mathcal{S}}\Big\{R_{\widehat{Z}^*_x|\mathcal{S}\cup\mathcal{O}}\Big\}\Big)\\
&= \max_{\mathcal{O}} \sum_{x\in\mathcal{O}} R_{\widehat{Z}^*_x|\mathcal{S}} + \sum_{x\in\mathcal{X}\setminus(\mathcal{S}\cup\mathcal{O})}\Big(R_{\widehat{Z}^*_x|\mathcal{S}} - \mathbb{E}_{Y_\mathcal{O}|y_\mathcal{S}}\Big\{R_{\widehat{Z}^*_x|\mathcal{S}}\Big\}\Big)\\
&= \max_{\mathcal{O}} \sum_{x\in\mathcal{O}} R_{\widehat{Z}^*_x|\mathcal{S}}\\
&= \sum_{i=1}^{k} \max_{x_i} R_{\widehat{Z}^*_{x_i}|\mathcal{S}} \ .
\end{aligned}$$

The first equality follows from $R_{\widehat{Z}^*_x|\mathcal{S}\cup\mathcal{O}} = 0$ for $x \in \mathcal{O}$ by assuming no observation noise. The second equality is due to the conditional independence assumption that is provided as a sufficient condition in the theorem. The third equality is due to the second summation term evaluating to zero. The last equality follows from the observation that each risk term in the summation depends only on the choice of the next location x to be observed by a single different robot. Hence, we can maximize each risk term in the summation independently and in a decentralized manner to achieve the same result as that in the third equality.

$$\begin{aligned}
&\max_x R_{\widehat{Z}^*_x|\mathcal{S}}\\
&= \max_x \{\min\left(P(Z_x = 1|y_\mathcal{S}), 1 - P(Z_x = 1|y_\mathcal{S})\right)\}\\
&= \max_x \{\min\left(P(Y_x \geq \gamma|y_\mathcal{S}), 1 - P(Y_x \geq \gamma|y_\mathcal{S})\right)\}\\
&\equiv \max_x \left\{\min\left[-\mathrm{erf}\left(\frac{\gamma-\mu_{x|\mathcal{S}}}{\sigma_{x|\mathcal{S}}\sqrt{2}}\right), \mathrm{erf}\left(\frac{\gamma-\mu_{x|\mathcal{S}}}{\sigma_{x|\mathcal{S}}\sqrt{2}}\right)\right]\right\}\\
&= \max_x -\left|\ \mathrm{erf}\left(\frac{\gamma-\mu_{x|\mathcal{S}}}{\sigma_{x|\mathcal{S}}\sqrt{2}}\right)\ \right|\\
&\equiv \min_x \left|\ \mathrm{erf}\left(\frac{\gamma-\mu_{x|\mathcal{S}}}{\sigma_{x|\mathcal{S}}\sqrt{2}}\right)\ \right|\\
&\equiv \min_x \frac{|\gamma-\mu_{x|\mathcal{S}}|}{\sigma_{x|\mathcal{S}}\sqrt{2}}\\
&\equiv \min_x \frac{|\gamma-\mu_{x|\mathcal{S}}|}{\sigma_{x|\mathcal{S}}} \ .
\end{aligned}$$

The first equality follows from (6). The first equivalence is due to $P(Y_x \geq \gamma|y_\mathcal{S}) = \frac{1}{2}\left[1 - \mathrm{erf}\left(\frac{\gamma-\mu_{x|\mathcal{S}}}{\sigma_{x|\mathcal{S}}\sqrt{2}}\right)\right]$.

Now, let us prove that the reward-maximizing strategy (12) for the entropy criterion is equivalent to to the DARE strategy (14). From (12),

$$\begin{aligned}
&\max_{\mathcal{O}}\ \mathbb{H}[Z_\mathcal{O}|y_\mathcal{S}]\\
&= \max_{\mathcal{O}} \sum_{x\in\mathcal{O}} \mathbb{H}[Z_x|y_\mathcal{S}]\\
&= \sum_{i=1}^{k} \max_{x_i} \mathbb{H}[Z_{x_i}|y_\mathcal{S}] \ .
\end{aligned}$$

The first equality follows from chain rule of entropy and conditional independence assumption. The second equality follows from observing that each entropy term in the summation depends only on the choice of the next location x to be observed by a single different robot. Hence, we can maximize each entropy term in the summation independently and in a decentralized manner to achieve the same result as that in the first equality.

$$\begin{aligned}
&\max_x\ \mathbb{H}[Z_x|y_\mathcal{S}]\\
&\equiv \min_x P(Z_x = 1|y_\mathcal{S}) \log P(Z_x = 1|y_\mathcal{S}) +\\
&\qquad (1 - P(Z_x = 1|y_\mathcal{S})) \log(1 - P(Z_x = 1|y_\mathcal{S}))\\
&\equiv \min_x \left|\frac{1}{2} - P(Z_x = 1|y_\mathcal{S})\right|\\
&= \min_x \left|\frac{1}{2} - P(Y_x \geq \gamma|y_\mathcal{S})\right|\\
&\equiv \min_x \left|\ \mathrm{erf}\left(\frac{\gamma-\mu_{x|\mathcal{S}}}{\sigma_{x|\mathcal{S}}\sqrt{2}}\right)\ \right|\\
&\equiv \min_x \frac{|\gamma-\mu_{x|\mathcal{S}}|}{\sigma_{x|\mathcal{S}}\sqrt{2}}\\
&\equiv \min_x \frac{|\gamma-\mu_{x|\mathcal{S}}|}{\sigma_{x|\mathcal{S}}} \ .
\end{aligned}$$

Robot Exploration with Fast Frontier Detection: Theory and Experiments

Matan Keidar
MAVERICK Group, Department of Computer Science, Bar-Ilan University
matankdr@gmail.com

Gal A. Kaminka
MAVERICK Group, Department of Computer Science, Bar-Ilan University
galk@cs.biu.ac.il

ABSTRACT

Frontier-based exploration is the most common approach to exploration, a fundamental problem in robotics. In frontier-based exploration, robots explore by repeatedly computing (and moving towards) *frontiers*, the segments which separate the known regions from those unknown. However, most frontier detection algorithms process the entire map data. This can be a time consuming process which slows down the exploration. In this paper, we present two novel frontier detection algorithms: *WFD*, a graph search based algorithm and *FFD* , which is based on processing only the new laser readings data. In contrast to state-of-the-art methods, both algorithms do not process the entire map data. We implemented both algorithms and showed that both are faster than a state-of-the-art frontier detector implementation (by several orders of magnitude).

General Terms

Algorithms Performance

Keywords

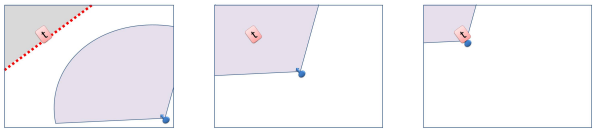
Robot, Exploration, Frontier, Laser

1. INTRODUCTION

The problem of exploring an unknown territory is a fundamental problem in robotics. The goal of exploration is to gain as much new information as possible of the environment within bounded time. The most common approach to exploration is based on *frontiers*. A frontier is a segment that separates known (explored) regions from unknown regions. By moving towards frontiers, robots can focus their motion on discovery of new regions. Yamauchi [17] was the first to show a frontier-based exploration strategy. His work preceded many others (e.g, [3, 4, 10, 11]).

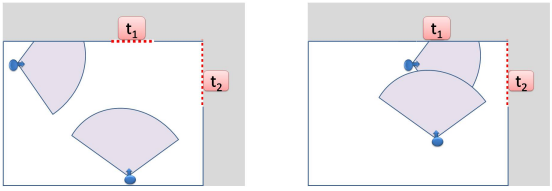
Most frontier detection methods are based on edge detection and region extraction techniques from computer vision. To detect frontiers, they process the entire map data with every execution to the algorithm. State-of-the-art frontier detection algorithms can take a number of seconds to run, even on powerful computers. If a large region is explored, the robot actually has to wait in its spot until the frontier detection algorithm terminates. Therefore, many exploration implementations call the frontier detection algorithm only when the robot arrives at its destination.

This can cause inefficiencies in the exploration. We present two examples: First, consider a common *single-robot case* (Figure 1), where a robot exploring its environment detects a frontier and moves towards it (Figure 1(a)). Because of sensor coverage, the robot may in fact sense (and clear) all remaining unknown area (Figure 1(b)), but because it cannot call the frontier-detection mechanism, it continues to move unnecessarily (Figure 1(c)). Similarly, consider a *multi-robot case* (Figure 2). Here, two robots, are exploring the environment, from their initial locations (Figure 2(a)). One of the robots passes by a target assigned to the other, thus clearing it (Figure 2(b)). But because the other robot cannot continuously re-detect frontiers, it unnecessarily continues towards the covered target, instead of turning to more fruitful exploration targets.



(a) (b) (c)

Figure 1: A single-robot example. In 1(a) the robot is heading towards the marked target on the frontier. In 1(b) the target and all of the remaining are covered by the robot's sensors, but because the robot does not re-detect frontiers, it continues to move. In 1(c) the robot has reached the frontier, unnecessarily.



(a) (b)

Figure 2: A multi-robot example. In 2(a), the top robot (R_2) is heading towards the right target, t_2; the other robot (R_1) heads towards the top target t_1. In 2(b) R_2 has reached its target, clearing both t_1 and t_2, making R_1's movements unnecessary.

In this paper, we thus focus on significantly speeding up frontier detection. We introduce two algorithms for fast frontier detection: The first, *WFD* (Wavefront Frontier Detector) is an iterative method that performs a graph-search over already-visited map points. It builds on ideas suggested in earlier work [5] which were not evaluated as an alternative to the edge-detection state-of-the-art. The key idea in *WFD* is that it does not scan the entire map, only the regions that have already been visited by the robot. However, as exploration progresses, the scanned area grows, and thus *WFD* cannot be expected to perform well in large areas. Our second contribution is *FFD* (Fast Frontier Detector), a novel approach for frontier detection which processes raw sensor readings, and thus only scans areas that could contain frontiers. But because it works with raw

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

sensor data, it requires extending the mapper (SLAM) with additional data-structures, so that frontiers are maintained even when they are no longer within sensor range. We describe these data-structures in detail, focusing on fast implementations.

We provide a detailed evaluation of these algorithms, and contrast them with the state-of-the-art (SOTA). We examine their performance in different types of environments and two different CPUs. We show that *WFD* is faster than SOTA by 1–2 orders of magnitude, and that *FFD* is faster than WFD by 1–2 orders of magnitude. The results make it possible to execute real-time frontier-detection on current-day robot CPUs, opening the way to novel frontier-based exploration methods which were impractical until now.

2. RELATED WORK

An outline of the exploration process can be described as follows: while there is an unknown territory, allocate each robot a target to explore and coordinate team members in order to minimize overlaps. In frontier-based exploration, targets are drawn from existing *frontiers*, segments that separate known and unknown regions (see Section 3.1 for definitions).

Most literature ignores the computational cost of frontier detection. To the best of our knowledge, all of the following works utilize a standard edge-detection method for computing the frontiers. They recompute frontier locations whenever one robot has reached its target location or whenever a certain distance has been traveled by the robots or after a timeout event.

Yamauchi [17] developed the first frontier-based exploration methods. The robots explore an unknown environment and exchange information with each other when they get new sensor readings. As a result, the robots build a common map (occupancy grid) in a distributed fashion. The map is continuously updated until no new regions are found. In his work, each robot heads to the *centroid*, the center of mass of the closest *frontier*. All robots navigate to their target independently while they share a common map. Frontier detection is performed only when the robot reaches its target.

Burgard et al. [3, 4] focus their investigation on probabilistic approach for coordinating a team of robots. Their method considers the trade-off between the costs of reaching a target and the utility of reaching that target. Whenever a target point is assigned to a specific team member, the utility of the unexplored area visible from this target position is reduced for the other team members. In their work, frontier detection is carried out only when a new target is to be allocated to a robot.

Wurm et al. [15] proposed to coordinate the team members by dividing the map into segments corresponding to environmental features. Afterwards, exploration targets are generated within those segments. The result is that in any given time, each robot explores its own segment. Wurm [16] suggests to call frontier detection every time-step of the coordination algorithm. Moreover, he claims that updating frontiers frequently is important in a multi-robot team since the map is updated not only by the robot assigned to a given frontier but also by all of the robots in the team. He suggests executing the algorithm $0.5m - 1m$ or every second or whenever a new target is requested.

Stachniss [12] introduced a method to make use of background knowledge about typical structures when distributing the team members over the environment. In his work, Stachniss computes new frontiers when there new targets are needed to be allocated. This happens whenever one robot has reached its designated target location or whenever the distance traveled by the robots or the elapsed time since last target assignment has exceeded a given threshold.

Berhault et al. [1] proposed a combinatorial auction mechanism where the robots bid on a bunch of targets to navigate. The robots are able to use different bidding strategies. Each robot has to visit all the targets that are included in his winning bid. After combining each robot's sensor readings, the auctioneer omits selected frontier cells as potential targets for the robots. Frontier detection is performed when creating and evaluating bids.

Visser et al. [14] investigated how limited communication range affects multi-robot exploration. They proposed an algorithm which takes into account wireless constraints when selecting frontier targets. Visser [13] suggests recomputing frontiers every 3–4 meters, which in his opinion, has positive effect.

Our work on *WFD* is independent from previous work, though [5] mentions a frontier detection algorithm that utilizes breadth-first search, similar to *WFD* . However, [5] does not provide details of the algorithm, nor evaluation of its performance, and so exact similarities and differences cannot be assessed. Our work here also significantly extends and corrects our own earlier work [9], which presented preliminary—and incomplete—versions of the *WFD* and *FFD* algorithms. Compared to [9], this paper presents corrected algorithms, proves the soundness and completeness of *FFD* , and reports new experimental and analytical results.

3. WAVEFRONT FRONTIER DETECTOR

We present a graph search based approach for frontier detection. The algorithm, *WFD* (Algorithm 1), processes the points on map which have already been scanned by the robot sensors and therefore, does not always process the entire map data in each run, but only the known regions.

3.1 Definitions and Terms

In this section we define and explain the terms that are used in the following sections. We assume the robot in question uses an occupancy-grid map representation in the exploration process (Figure 3) within the map:

Unknown Region is a territory that has not been covered yet by the robot's sensors.

Known Region is a territory that has already been covered by the robot's sensors.

Open-Space is a *known region* which does not contain an obstacle.

Occupied-Space is a *known region* which contains an obstacle.

Occupancy Grid is a grid representation of the environment. Each cell holds a probability that represents if it is occupied.

Frontier is the segment that separates known (explored) regions from unknown regions. Formally, a frontier is a set of *unknown* points that each have at least one *open-space* neighbor.

Definition. Suppose we are given a temporal sequence of observations $\langle O_0, \ldots, O_t \rangle$ (time 0 to time t), where each observation O_x is a tuple $\langle G_x, P_x, R_x \rangle$ composed of: (i) the occupancy-grid G_x of time x; (ii) the robot pose P_x (in occupancy-grid coordinates); and (iii) the range sensor readings R_x originating at the robot location (given in either ego-centric polar coordinates, or in occupancy-grid coordinates). The *Frontier Detection Problem* is to return all frontiers existing at time t, given the sequence.

Existing algorithms for frontier detection rely on edge-detection methods. The algorithms systematically search for frontiers all over the occupancy-grid, i.e., both in known and unknown regions.

3.2 WFD

WFD (Algorithm 1) is based on Breadth-First Search (BFS). First, the occupancy-grid point that represents the current robot position is enqueued into $queue_m$, a queue data-structure used to determine the search order (Lines 1–3).

Next, a BFS is performed (Line 4–30) in order to find all frontier points contained in the map. The algorithm keeps scanning only

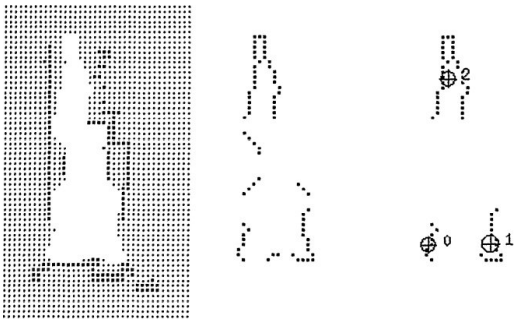


Figure 3: Evidence grid, frontier points, extraction of different frontiers (from left to right). Taken from [17].

Algorithm 1 Wavefront Frontier Detector (WFD)

Require: $queue_m$ // queue, used for detecting frontier points from a given map
Require: $queue_f$ // queue, used for extracting a frontier from a given frontier cell
Require: $pose$ // current global position of the robot

```
1:  queue_m ← ∅
2:  ENQUEUE(queue_m, pose)
3:  mark pose as "Map-Open-List"

4:  while queue_m is not empty do
5:    p ← DEQUEUE(queue_m)

6:    if p is marked as "Map-Close-List" then
7:      continue
8:    if p is a frontier point then
9:      queue_f ← ∅
10:     NewFrontier ← ∅
11:     ENQUEUE(queue_f, p)
12:     mark p as "Frontier-Open-List"

13:     while queue_f is not empty do
14:       q ← DEQUEUE(queue_f)
15:       if q is marked as {"Map-Close-List","Frontier-Close-List"} then
16:         continue
17:       if q is a frontier point then
18:         add q to NewFrontier
19:         for all w ∈ adj(q) do
20:           if w not marked as {"Frontier-Open-List","Frontier-Close-List", "Map-Close-List"} then
21:             ENQUEUE(queue_f,w)
22:             mark w as "Frontier-Open-List"
23:       mark q as "Frontier-Close-List"
24:     save data of NewFrontier
25:     mark all points of NewFrontier as "Map-Close-List"
26:   for all v ∈ adj(p) do
27:     if v not marked as {"Map-Open-List","Map-Close-List"} and v has at least one "Map-Open-Space" neighbor then
28:       ENQUEUE(queue_m,v)
29:       mark v as "Map-Open-List"
30:   mark p as "Map-Close-List"
```

points that have not been scanned yet and represent open-space (Line 27). The above scanning policy ensures that only known regions (that have already been covered by the robot's sensors) are actually scanned. The significance of this is that the algorithm does not have to scan the entire occupancy-grid each time.

Because frontier points are adjacent to open space points, all relevant frontier points will be found when the algorithm finishes (Line 30). If a frontier point is found, a new BFS is performed in order to extract its frontier (Lines 13–25). This BFS searches for frontier points only. Extracting the frontier is ensured because of the connectivity of frontier points. At the end of the extraction (Line 25), the extracted frontier data is saved to a set data-structure that stores all frontiers found in the algorithm run.

In order to avoid rescanning the same map point and detecting the same frontier reachable from two frontier points, *WFD* marks map points with four indications:

1. Map-Open-List: points that have already been enqueued by the outermost BFS (Line 28)
2. Map-Close-List: points that have already been dequeued by the outermost BFS (Line 5)
3. Frontier-Open-List: points that have already been enqueued by the frontier extraction BFS (Line 21)
4. Frontier-Close-List: points that have already been dequeued by the frontier extraction BFS (Line 14)

The above marks indicate the status of each map point and determine if there is a need to handle it in a given time.

The key innovation in *WFD* is that it prevents scanning unknown regions, since frontiers never appear there. However, it still searches all known space.

3.3 Speeding-Up WFD Even Further

WFD's execution time can be boosted even more by reducing the grid size. Of course, there is a trade-off between shorter execution time and the quality of the output frontiers. Even though, standard exploration tasks can utilize the output frontiers received in this manner. The grid is divided into blocks in size of the robot's width and height. Smaller blocks will not make sure that robot will be able to pass through terrain obstacles (i.e. corridors). Each block in the real world is represented by a single cell in the reduced grid. In order to determine the value of the cell, we examined different strategies. We considered both the speed of creating the new grid and the quality of the output. We found out that sampling the center of the block edges and the block center yields the best results.

4. FAST FRONTIER DETECTOR

Unlike other frontier detection methods (including *WFD*), our proposed algorithm (Algorithm 2) only processes new laser readings which are received in real time. It therefore avoids searching both known and unknown regions. In doing this, we make use of the fact that by definition, frontiers represent the boundaries between the known and unknown regions of the environment (see Figure 3). Hence, scanning all unknown regions is definitely unnecessary and not time-efficient. The *FFD* algorithm contains four steps (Algorithm 2), and can be called with every new scan.

4.1 Sorting

The first step (line 1) sorts range readings based on their angle, i.e., based on the ego-centric polar coordinates with the robot as the origin. Normally, laser readings are given as a sorted set of polar coordinated points, making this sorting step unnecessary. However, if this is not the case, a sorting is needed to be applied on the received laser readings.

To sort the readings, we assume that range readings are a set of Cartesian coordinated points, which consists of the locations

Algorithm 2 Fast Frontier Detector (FFD)

Require: $frontiersDB$ // data-structure that contains last known frontiers
Require: $pose$ // current global position of the robot
Require: lr // laser readings which were received in current iteration. Each element is a 2-d cartesian point

// polar sort readings according to robot position
1: $sorted \leftarrow SORT\_POLAR(lr, pose)$
// get the contour from laser readings
2: $prev \leftarrow POP(sorted)$
3: $contour \leftarrow \emptyset$
4: **for all** Point $curr \in sorted$ **do**
5: $line \leftarrow GET\_LINE(prev, curr)$
6: **for all** Point $p \in line$ **do**
7: $contour \leftarrow contour \cup \{p\}$
// extract new frontiers from contour
8: $NewFrontiers \leftarrow \emptyset$ // list of new extracted frontiers
9: $prev \leftarrow POP(contour)$
10: **if** $prev$ is a frontier cell **then** // special case
11: create a new frontier in $NewFrontiers$
12: **for all** Point $curr \in contour$ **do**
13: **if** $curr$ is not a frontier cell **then**
14: $prev \leftarrow curr$
15: **else if** $curr$ has been visited before **then**
16: $prev \leftarrow curr$
17: **else if** $curr$ and $prev$ are frontier cells **then**
18: add $curr$ to last created frontier
19: $prev \leftarrow curr$
20: **else**
21: create a new frontier in $NewFrontiers$
22: add $curr$ to last created frontier
23: $prev \leftarrow curr$
// maintainance of previously detected frontiers
24: **for all** Point $p \in ActiveArea$ **do**
25: **if** p is a frontier cell **then**
26: // split the current frontier into two partial frontiers
27: get the frontier $f \in frontiersDB$ which enables $p \in f$
28: $f_1 \leftarrow f[1 \dots p]$
29: $f_2 \leftarrow f[(p+1) \dots |f|]$
30: remove f from $frontiersDB$
31: add $f1$ and $f2$ to $frontiersDB$
32: **for all** Frontier $f \in NewFrontiers$ **do**
33: **if** f overlaps with an existing frontier $existFrontier$ **then**
34: $merged \leftarrow f \cup existFrontier$
35: remove $existFrontier$ from $frontiersDB$
36: add $merged$ to $frontiersDB$
37: **else**
38: create a new index and add f to $frontiersDB$

of range hits ($\{(x_0, y_0), \dots, (x_n, y_n)\}$ where n is the number of readings). The naive method for converting Cartesian to polar coordinates uses two CPU time-consuming functions: *atan2* and *sqrt*.

To speed angle sorting, we use a cross product [6] to avoid converting Cartesian to polar coordinates, while still sorting the points based on polar angle. Given 3 Cartesian coordinated points:

$$P_0 = (x_0, y_0), P_1 = (x_1, y_1), P_2 = (x_2, y_2)$$

the *cross product* is defined as:

$$(p_1 - p_0) \times (p_2 - p_0) = (x_1 - x_0) \cdot (y_2 - y_0) - (x_2 - x_0) \cdot (y_1 - y_0)$$

If the result is positive, then $\overrightarrow{P_0P_1}$ is clockwise from $\overrightarrow{P_0P_2}$. Else, it is counter-clockwise. If the result is 0, then the two vectors lie on the same line in the plane (i.e., the angle is the same).

Therefore, by examining the sign of the cross product, we can determine the order of the Cartesian points according to polar coordinates, without calculating their actual polar coordinate value. This applies only five subtractions and two multiplications which are far less time-consuming than calling *atan2* and *sqrt*.

4.2 Contour

In this step (lines 2–7) we use the angle-sorted laser readings. The output of the contour step is a contour which is built from the sorted laser readings. The algorithm computes the line that lies between each two adjacent points from the set. The line is computed by calling the function GET_LINE. In our implementation we use *Bresenham's line algorithm* [2]. Next, all points that are represented by all the lines (including the points from the laser readings set) are merged into a contour (Figure 4).

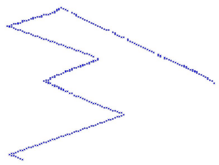


Figure 4: Example of produced contour.

4.3 Detecting New Frontiers

In this step (lines 8–23) the algorithm extracts new frontiers from the previously calculated contour. There are three cases correspond to each two adjacent points in the contour:

1. **Current scanned point is not a frontier cell:** therefore, it does not contribute any new information about frontiers and can be ignored.
2. **Current and previous scanned points are frontier cells:** therefore, both points belong to the same frontier and current scanned point is added to last detected frontier.
3. **Current point is a frontier cell but the previous is not:** a new starting point of a frontier was detected. Hence, the algorithm creates a new frontier and adds the new starting point to it.

4.4 Maintaining Previously Detected Frontiers

FFD gains its speed by processing the laser readings only, rather than entire regions of the map. However, if the robot navigates towards a specific frontier, other previously detected frontiers are no longer updated because they are not covered by the robot's sensors. Thus, scanning the new received laser readings enables *FFD* to detect only *new* frontiers in each execution. In this step (lines 24–38), in order to get complete information about the frontiers, the algorithm performs maintenance over previously detected frontiers which are no longer covered in the range of the sensors. Only by joining together new detected frontiers and previously detected frontiers, we get the overall frontiers in current world state. This step has multiple targets: avoiding detection of new frontiers in

an already scanned area (Section 4.4.2), eliminating frontier points which are no longer belong to frontiers (Section 4.4.3) and joining correctly the new detected frontiers together with previously detected frontiers (Section 4.4.4).

4.4.1 Data-Structures

In order to perform the maintenance step within a very short time as possible, *FFD* utilizes two data-structures which have a short access time. These data structures must maintain memory of frontiers between calls. Thus *FFD* has to have persistent memory, i.e., data structures that persist between calls. This is contrast to other approaches that can be executed in a certain time, and only then.

Another thing to note is that in particle filter based systems (our focus in this paper), each particle represents a possible hypothesis of the world state (including the robot position of course). The "best" particle is chosen according to a likelihood measurement. *FFD* requires the previously detected frontiers to be robust against map orientation changes caused by loop-closures of the mapping algorithm. Therefore, when a new laser reading is received, each particle executes its own instance of *FFD* algorithm on its own map, using its own data structures. More specifically, each particle performs maintenance with its own map because particles do not share maps. We describe the data structures for maintenance below.

Grid of Frontier Indices This data-structure is an extension of the occupancy grid (though it can be implemented as a separate entity). In addition to occupancy information, each grid cell contains *a frontier index*, pointing to a frontier to which the grid cell belongs, or NULL otherwise. The pointer is into the Frontier Database (described below). In our implementation, we used integer index values. After accessing a grid cell, querying for its frontier index is $O(1)$.

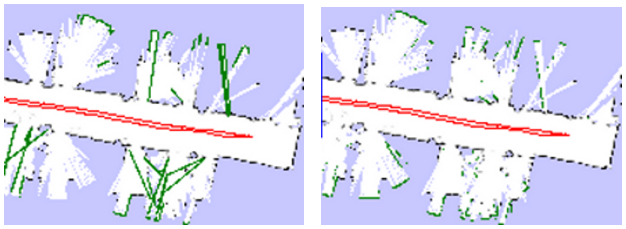
Frontier Database This data-structure maps frontier indexes (pointers) to sets of points. All detected frontiers are stored in this data-structure. We use it to map frontier index to the actual set that contains the points in world coordinates. In our implementation, we use the default C++ implementation of a map template, which is implemented as a self-balancing binary search tree. Therefore, assuming n represents the number of frontiers stored in the database, searching for a frontier index takes $O(\log n)$, inserting a new frontier takes $O(\log n)$ and removing a frontier index takes $O(\log n)$, though a (hash) table lookup implementation can make this faster.

4.4.2 Avoiding Re-Detection of Same Frontier

FFD detects new frontiers by processing laser readings only. Hence, *FFD* might detect the same frontier again and classify it as a new frontier if the robot did not change its position during two following *FFD* executions. Moreover, if the robot travels back to an already visited region, no new frontiers should be detected. *FFD* has to distinguish between laser readings from between time frames. Otherwise, *FFD* might wrongly detect a new frontier which lies within an already scanned area.

Therefore, we keep track of the number of sensor visits (sensor covers) of each map cell. The definition of a frontier point is now expanded: a frontier point is a point which represent an unknown region, has at least one open-space neighbor and has not been scanned before. Given a contour, the detection of new frontiers ignores points that have already been scanned by the laser sensors and treats them as non-frontier points (lines 15–16).

Figure 5 demonstrates the necessity of the above. Figure 5(a) shows frontiers extraction without tracking the number of visits. It can be seen that there are frontiers that lie inside an open space



(a) Incorrectly re-detected frontiers. (b) Correct detection.

Figure 5: An example of re-detecting same frontiers.

area. This is absolutely wrong because frontiers are supposed to be positioned on the boundaries between known and unknown regions. In contrast, Figure 5(b) shows frontiers extraction with avoidance of redetecting same frontiers. It can be seen that every frontier separates known and unknown regions.

4.4.3 Eliminating Previously Detected Frontiers

In order to complete the process, points which are no longer in frontiers (i.e. were covered by the robot's sensors) have to be eliminated. Lines 24–31 contain the elimination logic applied by *FFD*.

Let t_i be a time frame and lr_{t_i} be the laser readings which were received in time frame t_i. In order to perform maintenance in a specific time, we define the *Active Area* of time frame t_i to be the blocking rectangle that can be constructed using the farthest laser readings of lr_{t_i}, relative to the robot position in time frame t_i.

$$x_{min} = min(\{x|x \in lr_{t_i}\}), y_{min} = min(\{y|y \in lr_{t_i}\})$$

$$x_{max} = max(\{x|x \in lr_{t_i}\}), y_{max} = max(\{y|y \in lr_{t_i}\})$$

$$ActiveArea_{t_i} = \{(x,y)\,|x_{min} \leq x \leq x_{max}, y_{min} \leq y \leq y_{max}\}$$

The active area's rectangle is constructed from the following vertices: (x_{min}, y_{min}), (x_{min}, y_{max}), (x_{max}, y_{max}), (x_{max}, y_{min}). The rectangle is an approximation to the real active area that is actually bounded within the laser readings.

By processing received laser readings, *FFD* extracts new frontiers. However, in order to get the complete world's frontiers state, points that are no longer on frontiers have to be eliminated. *FFD* maintains a frontier database which maps an integer (frontier index) to a set of points (frontier).

An unknown region is classified as known region only if it is covered by the robot's sensors. *FFD* gets its input from the new received laser readings, and thus only regions that are covered by the robot's sensors might contain frontiers that have to be eliminated. Thus, if there are frontiers that need to be eliminated, they must lie inside the *Active Area*. Hence, the active area is a key feature in the process of maintaining frontiers. *FFD* scans each point that lies inside the active area and checks if it was previously belonged to a frontier. The check can be performed very fast as explained before. If the current scanned point was belonged to a frontier, the current scanned point is removed from the frontier and the frontier is spit into two partial frontiers using the current scanned point as a pivot (lines 28–29).

In the end of this process, all no-longer frontier points in the frontier database are removed and the database contains only points that are still valid frontiers.

4.4.4 Storing New Detected Frontiers

In the last phase of the maintenance step (lines 32– 38) new detected frontiers are stored in the frontier database alongside with existing valid frontiers. For each new detected frontier, *FFD* checks if it overlaps with an already existing frontier. This comparison can be performed in a short time using the matrix of frontier indices.

Each frontier point is queried in $O(1)$ operations. If an overlap is found, the frontier is merged with the frontier that it is overlapped with. If no overlap is found, then the frontier is inserted to the frontier database.

5. FFD IS SOUND AND COMPLETE

We show that Algorithm 2 is sound and complete. We begin with a lemma that demonstrates that *FFD* always recognizes new frontiers (i.e., frontiers that appeared at time t, but did not exist before). This will then be used to prove completeness of *FFD* .

LEMMA 5.1. *Suppose f is a frontier point at time t, which was not a frontier point at any time s, where $s < t$. Then FFD will mark f as a frontier given observation O_t.*

PROOF. Let f be a valid frontier point in time t and was not a classified as frontier in time $s < t$. Since f is a valid frontier point, then it has a value of *Unknown* and has at least one *Open Space* neighbor at time t. Assume towards a contradiction that *FFD* did not recognize f as a frontier point. First, let us show that f is contained in the contour handled in Lines 8–23. Since f is a valid frontier point, then it has a value of *Unknown* and has at least one *Open Space* neighbor in time t. The point f cannot be located wholly within an *unknown* region because it must have at least one *Open Space* neighbor. Also, the point f cannot be located wholly within a *known* region since f is a valid frontier point and hence, its value is $Unknown$. Therefore, f must be located on the contour itself. Lines 8–23 handle points on the contour, which we have just shown f is on. In these lines, the *FFD* algorithm scans *all* contour points sequentially and specifically searches for frontier points. Because if scans *all* points on the contour, and we have shown that f is on the contour, it follows that f would be detected, contradicting the assumption that *FFD* did not recognize f as a frontier point at time t. □

We now turn to proving the completeness of the *FFD* algorithm.

THEOREM 5.2. *Let f be a valid frontier point at time t. Then* FFD *will mark f as a frontier point given the sequence of observations $\langle O_0, \ldots, O_t \rangle$.*

PROOF. Two cases should be examined:

Case 1. f is a new frontier point at time t. Trivially, this case is handled directly by lemma 5.1.

Case 2. f was a new frontier point at time s, where $s < t$. Let s be the earliest time in which f was a frontier. Based on lemma 5.1, it follows that it was detected at this time. All that remains to show is that given the f is still valid at time t, *FFD* will maintain knowledge of it from time s and report on it.

If f is still a valid frontier point at time t, then it has not been covered yet by the robot's sensors. Otherwise, it would no longer contain an *Unknown* value and hence, will not be a valid frontier point. So if it was not yet covered, it must be a frontier point that is maintained by *FFD*. The only way in which f can be eliminated from being classified as a frontier point is done by lines 24–31. In these, *FFD* scans all points that are covered by the robot's sensors and checks if any points should be eliminated (line 25). Since f is not covered by the sensors, then it will not be scanned and eliminated in time $t \Rightarrow f$ remains classified as a frontier by *FFD*.

In both cases we show *FFD* will recognize f to be a valid frontier at time t. □

Since Theorem 5.2 is true for any frontier point valid at time t, it follows that *FFD* is complete.

To show the soundness of *FFD* , we must demonstrate that there does not exist a case where *FFD* marks a point $\hat{f}$ as a frontier, when it is not.

(a) Cartesium Building, University of Bremen. (b) Freiburg, Building 079.

Figure 6: Some of the testing environments.

THEOREM 5.3. *Let $\hat{f}$ be an arbitrary point in the occupancy grid, which is* not *a frontier at time t. Then* FFD *will not return $\hat{f}$ as a frontier point, given the sequence of observations $\langle O_0, \ldots, O_t \rangle$.*

PROOF. Assuming that $\hat{f}$ is an arbitrary point which is *not* a frontier point at time t, then $\hat{f}$ is either contains value different from *Unknown* or all its adjacent values are different from *Open Space*. We will examine two cases:

Case 1. $\hat{f}$ is marked as a new frontier. Suppose, towards a contradiction, that *FFD* detects $\hat{f}$ as a new frontier (i.e., true at time t, but not a frontier in time s, where $s < t$). Since detection of new frontier points (Lines 8–23) considers only points on the contour, it follows that $\hat{f}$ must be located on the contour *and* detected by lines 8–23. However, line 13 specifically avoids classifying non-frontier points as frontiers. Since $\hat{f}$ is a non-frontier point, it is ignored by FFD. Therefore, $\hat{f}$ cannot be marked as a new frontier $\Rightarrow$ contradicting the assumption that it is detected by FFD as a new frontier. Case 1 is impossible.

Case 2. $\hat{f}$ is an old frontier but was not eliminated by the maintenance routine. Suppose, towards a contradiction, that $\hat{f}$ is located inside the active area and is not eliminated by the maintenance section. Therefore, $\hat{f}$ is a point that was covered by the robot's sensors and no longer contains an *Unknown* value, yet is still marked as a frontier by the FFD algorithm. We remind the the reader that in order to maintain frontier points across runs, each point in the grid keeps a value which contains NULL if the point is not a frontier point or the index of the frontier to whom it belongs. Therefore, in line 25 FFD scans all points in the active area and checks if they contain a frontier index. When FFD scans $\hat{f}$, it will find out that it contains a valid frontier index (because it has previously belonged to a valid frontier) and continues executing lines 27–31. In these lines, FFD checks and removes from the DB all points that are no longer frontier points and previously were frontier points. Thus $\hat{f}$ will be eliminated after scanning the active area, contradicting the assumption that $\hat{f}$ was not eliminated. □

Since in both cases we show that FFD necessarily eliminated $\hat{f}$ from the valid frontier list, it follows that if $\hat{f}$ is not a frontier-point at time t, it would not be marked as such by FFD. Since Theorem 5.3 holds for any arbitrary point, it follows that FFD never incorrectly marks a non-frontier point as a frontier. It is thus sound.

6. EXPERIMENTAL RESULTS

We have fully implemented *WFD* and *FFD* and performed testings on data obtained from the Robotics Data Set Repository (Radish) [8]. We used *WFD* without the suggested speed-up feature, in order to compare all algorithms fairly. Figure 6 shows a few of the environments used for the evaluation. *WFD* and *FFD* were compared with a *SOTA* (state-of-the-art) frontier detection algorithm, due to Wurm and Burgard [15, 16].

To evaluate the algorithms, we integrated them into a single-robot exploration system. The system is based on GMapping, an open-source SLAM implementation [7]. We integrated our code

into the *ScanMatcher* component which is contained inside *gsp thread* (Grid SLAM Processor). At the time that a new MapEvent is raised, all frontier detection algorithms are executed according to current world state. Execution times are measured by Linux system-call *getrusage*, which measures the CPU-process time. We examined the run-time of all algorithms on two different machines:

- First experiment: we used a fast desktop computer containing Intel Core 2 Duo T6600 CPU with clock speed of 2.20 GHz and Random Access Memory (RAM) in size of 4 GB.
- Second experiment: we used a slower desktop computer containing Intel Pentium III (Coppermine) with clock speed of 800 MHz and Random Access Memory (RAM) in size of 1 GB. Research-grade robots typically have a faster CPU, but commercial robots typically do not.

We used several environments taken from Radish [8]:

(A) Cartesium Building, University of Bremen
(B) Freiburg, Building 079
(C) Outdoor dataset recorded at the University of Freiburg, (C)
(D) 3rd Floor of MIT CSAIL
(E) Edmonton Convention Centre (site of the AAAI 2002 Grand Challenge)

Note that we use the exploration data (raw sensor readings and odometry) from these data sets, and thus all algorithms use exactly the same data, form the same robot trajectories. Thus the movement of the robot is identical, and the only thing we examine is how quickly it can compute frontiers.

FFD is called every-time a new laser reading is received. Therefore, in order to compare *FFD* execution time to other algorithms correctly, we accumulate *FFD*'s execution times between calls to other algorithms. In other words, if we call *WFD* in time-stamps t_i and t_{i+1}, then *FFD*'s accumulated execution time is calculated by:

$$\sum_{x=t_i}^{t_{i+1}} ExecutionTime_{FFD}(x)$$

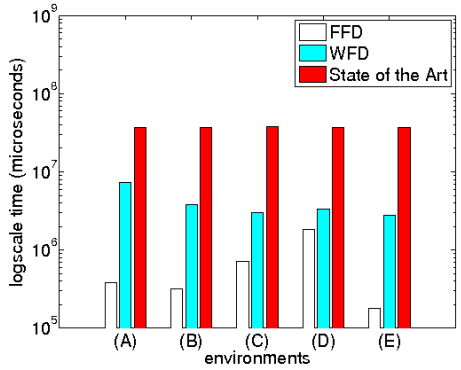
Moreover, we remind the reader that because *FFD* is called for every particle in the particle-filtering GMapping [7], the results here accumulate also over the number of particles (30 in our case).

We begin by examining overall performance. Figure 7 shows one set of results of the comparison in each of the two machines. Each group of bars represents a run over a separate map. For each algorithm, we calculate the mean execution time, over the duration of the exploration. The vertical axis measures the calculated execution time in microseconds, on a *logarithmic scale*. The one-second line is at 10^6 microseconds. The next tick, at 10^7, marks 10 seconds.

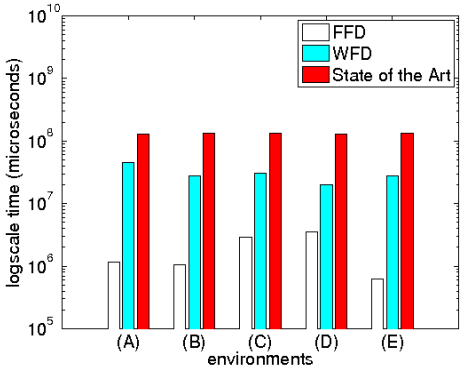
Figure 7 shows that *WFD* is faster than *SOTA* by approximately one order of magnitude. *FFD* is faster than *WFD* by one to two orders of magnitude. Indeed, *FFD* performs close to the one-second line. In contrast, *WFD* and *SOTA* typically take anywhere from 10 to 100 seconds to perform their task, even on relatively fast machines.

FFD's improvement over the others is indeed notable, given that the measured results are not for single *FFD* runs, but in fact show accumulated run-time, over the frequency of the sensor readings, multiplied over the number of particles (approximately 2000 calls to *FFD* for each *WFD* or *SOTA* calls). These multiplicative factors have significant impact on *FFD*'s usability. It is important to understand whether the number of particles influences the result more than the frequency of sensor readings, as the number of particles is often increased for better quality.

We thus turn to evaluating *FFD* at a finer resolution. Figure 8 compares the run-time of individual particles in specific environments. Each bar represents a specific particle. The vertical axis



(a) Intel T6600



(b) Intel Coppermine

Figure 7: Comparing WFD and FFD to State-of-the-Art algorithm on different machines.

measures the mean run-time of *FFD* for the particle. The error-bars represent the standard deviation of each particle's run-time.

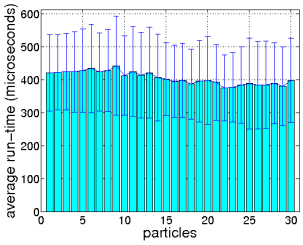
The figure shows that the per-particle run-time is measured in a few hundred micro-seconds. Thus the overall results were accumulating comparing the accumulation of thousands of *FFD* runs against single *WFD* and *SOTA* runs. Indeed, one can boost *FFD*'s execution time by not executing it on every received laser reading, since the frequency of receiving new laser readings is often higher than the speed of processing and updating the map anyways. Many laser sensors generate output at 30Hz–75Hz, at least three times faster than the rate at which the robots process the information. By ignoring some laser readings, *FFD* would perform much better, without any noticeable decay in mapping quality.

7. CONCLUSIONS AND FUTURE WORK

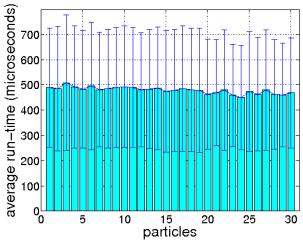
Frontier-based exploration is the most common approach to solve the exploration problem. State-of-the-art frontier detection methods process the entire map data, which hangs the exploration system for a few seconds with every call to the detection algorithm.

We introduced two novel faster frontier detectors, *WFD* and *FFD*. The first, a graph based search, processes the map points which have already been scanned by the robot sensors and therefore, does not process unknown regions in each run (though it grows slower as more area is known). The second, a laser-based approach for frontier detection, only processes new laser readings which are received in real time eliminating also much of the known area search. However, maintaining previous frontiers knowledge requires tight integration with the mapping component, which may not be straightforward. We describe efficient implementation for both algorithms, and compare them empirically. *FFD* is shown to outperform *WFD* and the state-of-the-art by 1–2 (2–3, resp.) orders of magnitude.

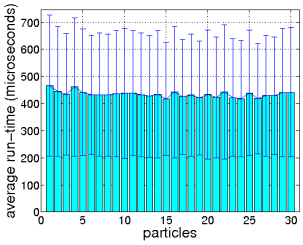
In future, we plan to integrate *FFD* with EKF-based SLAM mappers, which we hope will lead to further improvements. We also



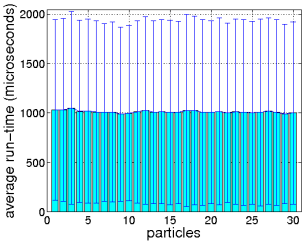
(a) Cartesium Building, Bremen



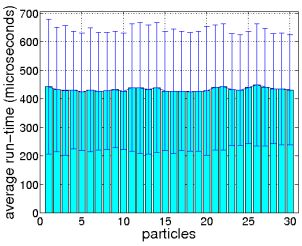
(b) Freiburg, Building 079



(c) Outdoor dataset, University of Freiburg



(d) 3rd Floor of MIT CSAIL



(e) Edmonton Convention Centre

Figure 8: *FFD* run-time for individual SLAM particles.

plan to begin investigation of novel exploration policies, based on real-time frontier-detection.

Acknowledgements. We thank Kai M. Wurm and Wolfram Burgard for providing us with their own implementation of state-of-the-art frontier detection algorithm. Thanks go to Cyrill Stachniss, Giorgio Grisetti and Nick Roy for providing data to the Robotics Data Set Repository (Radish) [8].

8. REFERENCES

[1] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt. Robot exploration with combinatorial auctions. In *IROS-03*, pages 1957–1962, 2003.

[2] J. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 2010.

[3] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun. Collaborative multi-robot exploration. In *IEEE International Conference on Robotics and Automation. Vol. 1*, pages 476–481, 2000.

[4] W. Burgard, M. Moors, C. Stachniss, and F. Schneider. Coordinated multi-robot exploration. *IEEE Transactions on Robotics*, 21(3):376–378, 2005.

[5] D. Calisi, A. Farinelli, L. Iocchi, and D. Nardi. Multi-objective exploration and search for autonomous rescue robots: Research articles. *J. Field Robot.*, 24:763–777, August 2007.

[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2001.

[7] G. Grisetti, C. Stachniss, and W. Burgard. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *IEEE Transactions on Robotics*, 23:34–46, 2007.

[8] A. Howard and N. Roy. The robotics data set repository (RADISH), 2003.

[9] M. Keidar, E. Sadeh-Or, and G. A. Kaminka. Fast frontier detection for robot exploration. In F. Dechesne, H. Hattori, A. ter Mors, J. M. Such, D. Weyns, and F. Dignum, editors, *Advanced Agent Technology: AAMAS 2011 Workshops. Revised Selected Papers*, volume 7068 of *Lecture Notes in Computer Science (LNCS)*, pages 281–294. 2012.

[10] H. Lau and A. NSW. Behavioural approach for multi-robot exploration. In *Australasian Conference on Robotics and Automation (ACRA), Brisbane, December*, 2003.

[11] R. Sawhney, K. M. Krishna, and K. Srinathan. On fast exploration in 2D and 3D terrains with multiple robots. In *AAMAS-09*, pages 73–80, 2009.

[12] C. Stachniss. *Exploration and Mapping with Mobile Robots*. PhD thesis, University of Freiburg, Department of Computer Science, 2006.

[13] A. Visser. personal communication. Email, January 4th, 2011.

[14] A. Visser and B. A. Slamet. Including communication success in the estimation of information gain for multi-robot exploration. In *WiOpt-08*, pages 680–687, 2008.

[15] K. Wurm, C. Stachniss, and W. Burgard. Coordinated multi-robot exploration using a segmentation of the environment. In *IROS-08*, Nice, France, Sept. 2008.

[16] K. M. Wurm. personal communication. Email, January 20th, 2011.

[17] B. Yamauchi. Frontier-based exploration using multiple robots. In *Agents-98*, pages 47–53, 1998.

Dynamic Reconfiguration in Modular Robots using Graph Partitioning-based Coalitions

Prithviraj Dasgupta, Vladimir Ufimtsev
Computer Science Department
University of Nebraska, Omaha, USA
{pdasgupta, vufimtsev}@unomaha.edu

Carl Nelson, S. G. M. Hossain
Mechanical Engg. Department
University of Nebraska, Lincoln, USA
cnelson5@unl.edu, smgmamur@yahoo.com

ABSTRACT

We consider the problem of dynamic self-reconfiguration in a modular self-reconfigurable robot (MSR). Previous approaches to MSR self-reconfiguration solve this problem using algorithms that search for a goal configuration in the MSR's configuration space. In contrast, we model the self-reconfiguration problem as a constrained optimization problem that attempts to minimize the reconfiguration cost while achieving a desirable configuration. We formulate the MSR self-reconfiguration problem as finding the optimal coalition structure within a coalition game theoretic framework. To reduce the complexity of finding the optimal coalition structure, we represent the set of all robot modules as a fully-connected graph. Each robot module corresponds to a vertex of the graph and edge weights represent the utility of a pair of modules being in the same coalition (or, connected component). The value of a coalition structure is then defined as the sum of the weights of all edges that are completely within the same coalition in that coalition structure. We then use a graph partitioning technique to cluster the vertices (robot modules) in the constructed graph so that the obtained coalition structure has close to optimal value. The clustering algorithm has time complexity polynomial in the number of agents, n, and yields an $O(\log n)$ approximation. We have verified our technique experimentally for a variety of settings. Our results show that the graph clustering-based self-reconfiguration algorithm performs comparably with two other existing algorithms for determining optimal coalition structures. [1]

Categories and Subject Descriptors

I.2.9 [**Robotics**]: Autonomous vehicles—*modular robots, dynamic reconfiguration*

General Terms

Algorithms

[1]This research has been performed as part of the ModRED project that is supported by a NASA EPSCoR grant.

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

Keywords

modular self-reconfigurable robots, dynamic reconfiguration, coalition game, graph-based clustering

1. INTRODUCTION

Over the past few years, modular self-reconfigurable robots (MSRs) have been proposed as an elegant, yet efficient way to build robots that are capable of maneuvering in tight spaces or unstructured terrain [18]. Structurally, an MSR is composed of functionally simple modules that are connected together into a certain formation. Each module is individually capable of performing very limited operations, but when connected with other modules, they can adapt their shape to form a single robot that can accomplish a complex task. In spite of the simple and inexpensive construction of an MSR's modules, and easy maneuverability, a principal challenge in MSRs is to solve the self-reconfiguration problem i.e. how to adapt their shape autonomously so that they can change tasks or continue their operation after encountering obstacles or occlusions that impede their movement. As a motivating example, we consider a scenario where a set of robot modules are deployed individually, possibly scattered on the ground within communication range of each other, from an airborne vehicle. The objective of these individual modules is to autonomously determine suitable multi-module configurations, maneuver themselves to get within close proximity of each other, and, finally align and dock with each other to realize those configurations. This paper focuses on the computational aspects of the problem faced by the individual modules to determine their 'best' set of configurations that gives them an improved efficiency or value in performing their assigned task, while considering the costs in terms of energy expended to get in proximity of, and align and dock with each other to get into those configurations. This problem is challenging because a fixed set of rules does not work for all situations. An MSR needs to perceive its current environment to determine how many modules to connect together, and the configuration or shape those modules should get into, so that the MSR can perform its assigned task most efficiently.

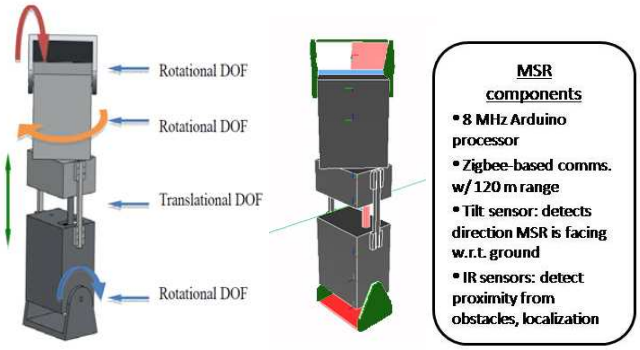
In this paper, we have addressed the MSR self-reconfiguration problem by modeling it as a coalition structure generation (CSG) problem in coalition game theory. Coalition games are suitable for the MSR self-reconfiguration problem because the solution found by a coalition game ensures stability. Once the best partition or coalition of agents, corresponding to the best configuration of MSRs has been found, the MSR modules that have been determined to form the

new configuration will remain together and will not try to leave the new configuration and attempt to combine with other modules. However, there are several research challenges that need to be addressed while using coalition game theory for MSR self-reconfiguration. First, in coalition game theory, the assimilation of agents into teams and the communication between agents is assumed to be free of cost. However, for MSRs, modules incur "cost" by expending energy to communicate with each other and physically move to each other's proximity to dock with each other. Secondly, solving the CSG problem that deals with finding the *best* or optimal coalition in a coalition structure graph is known to be an NP-hard problem with a few existing heuristic solutions. To address these problems, in this paper we first develop a utility-based formulation for the costs corresponding to the dynamic reconfiguration problem in MSRs within a coalition game theoretic framework. Then we use a graph clustering algorithm to solve the CSG problem within this setting, using a polynomial time complexity and logarithmic approximation. To illustrate the operation of our MSR we have used the domain of robotic exploration of initially unknown environments. Our experimental results show that our graph clustering technique can be successfully used to dynamically self-reconfigure an MSR into different configurations.

2. RELATED WORK

Modular self-reconfigurable robots (MSRs) are a type of self-reconfigurable robots that are composed of identical modules. These modules can change their connections with each other to manifest different shapes of the MSR and select a shape that enables the MSR to perform its assigned task efficiently [4, 16]. An excellent overview of the state of the art MSRs and related techniques is given in [18]. Out of the three types of MSRs — chain, lattice and hybrid - we have used a chain-type MSR to illustrate the experiments in this paper although our techniques could be used for other types too. The self-reconfiguration problem in MSRs has been solved using search-based [3, 5] and control-based techniques [14]. However, both these techniques require the initial and goal configuration to be determined before the reconfiguration process starts. A third technique called task-based reconfiguration has recently shown considerable success [9]. Here the goal configuration of an MSR doing reconfiguration is not determined *a priori*, but is determined as the configuration that helps the MSR perform its task efficiently. Our work in this paper is targeted towards task-based reconfiguration techniques; we do not explicitly specify a goal configuration but allow the reconfiguration algorithm to select a new configuration that minimizes the reconfiguration cost.

Coalition game theory gives a set of techniques that can be used by a group of agents to form teams or coalitions with each other [10, 13]. A coalition can be loosely defined as a set of agents that remain together with the intention of cooperating with each other, possibly to perform a task. In terms of MSRs a coalition represents a set of MSR-modules that are connected together while performing a certain task. Within coalition games, the coalition structure generation problem that deals with partitioning the agents into disjoint and exhaustive sets called coalitions has received significant attention. This problem is NP-complete, and Sandholm [15] and Rahwan [11] have proposed anytime algorithms to find



(a) (b) (c)

Figure 1: (a) CAD figure of a single module of the MSR. (b) A simulated version of the MSR inside Webots. (c) Major components of the MSR.

near-optimal solutions. In contrast to these works, we use a graph clustering-based approach to find the optimal coalition structure.

Weighted graph games were introduced in [8] as a specific case of coalition games where the set of agents is modelled as a vertex set of a graph, and the valuation function is calculated by summing the edge weights of the constructed graph. Coalition structure generation in graph games has recently received attention in [17], [1], and the problem was shown to be NP-complete. Our method uses the graph coalition game formulation and the graph clustering technique presented in [7] to obtain close to optimal coalition structures in the graph. The technique is based on a generalized version of correlation clustering [2] known as correlation clustering with partial information.

3. A NOVEL 4-DOF MODULAR ROBOT

We have used an MSR called ModRED [6] that is currently being developed by us, for implementing and testing the techniques in this paper. Unlike most other MSRs, it has 4 DOF (3 rotational and 1 translational); this allows each module to rotate along its long axis as well as extend along that same axis, as shown in Figure 1(a). This combination of DOF enables the MSR to achieve a greater variety of gaits to possibly maneuver itself out of tight spaces. A picture of the MSR, its simulated version within a robot simulator called Webots and its major components are shown in Figure 1. For the simulated version of each module, we have used a GPS node that gives global coordinates on each robot[2], an accelerometer to determine the alignment of the robot with the ground, in addition to the IR sensors and Zigbee modules in the physical robot. The movement of the MSR in fixed configuration is enabled through gait tables [16]. Each gait table applies to a specific movement of the robot in a specific configuration. The contents of the gait table give the sequence of movements of the different joints of the robot to achieve the desired motion. Videos showing the movement of the MSR in different configurations using gait tables are available at http://cmantic.unomaha.edu/projects/modred/. The MSR ModRED can be configured into a chain struc-

[2]In the physical MSR, relative positioning is planned to be done using the IR sensors.

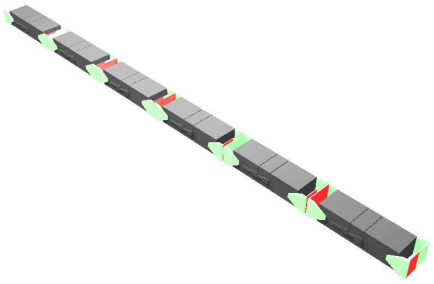


Figure 2: ModRED modules in a chain configuration

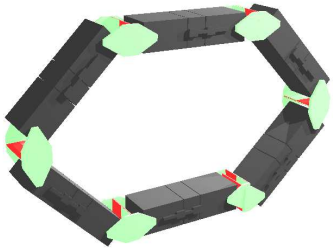


Figure 3: ModRED modules in a ring configuration

ture as is shown in Figure 2 as well as a ring structure as is shown in Figure 3 (images taken from Webots). When modules form these configurations the MSR performs more efficiently; it can move faster and it can overcome obstacles. In different configurations, the MSR uses different gaits which enable it to move faster and overcome more obstacles. In the chain structure the MSR can mimic the movement of a snake for movement or it can use its rotational degree of freedom to roll sideways, while in the ring structure the MSR can perform several "rolling" motions akin to that of a wheel.

While moving in a fixed configuration, if the MSR's motion gets impeded by an obstacle or an occlusion in its path, it needs to reconfigure into a new configuration so that it can continue its movement efficiently. In the next section, we formalize the MSR self-reconfiguration problem and then provide a graph clustering approach for finding the optimal coalition.

4. DYNAMIC SELF-RECONFIGURATION IN MSRS

Let A be the set of modules or agents that have been deployed in the environment. The set of MSRs (coalition structure) at time t, $\{A_i^t\}$, is defined as a set of exhaustive and disjoint partitions of A, i.e., $\cup_i A_i^t = A$ and $A_i \cap A_j = \emptyset$ for $i \neq j$. The i-th MSR (coalition) at time t is given by a set of ordered modules or agents, i.e.,

$$A_i^t = \{a_{i_1}^t, a_{i_2}^t, a_{i_3}^t, ..., a_{i_{|A_i^t|}}^t\} \quad (1)$$

Here, $a_{i_1}^t$ is the leading module of A_i^t, $a_{i_{|A_i^t|}}^t$ is the trailing or end module of the MSR and $\{a_{i_j}^t, a_{i_{j+1}}^t\}, j = 1...|A_i^t| - 1$ are the set of modules that are physically coupled together pairwise in a chain configuration using their end couplers. Using this definition, when A_i^t is a singleton, it represents a single module that is not coupled with any other modules.

Let $\Pi(A)$ be the set of all partitions of A and let $CS(A) = \{A_1, A_2, ..., A_k\} \in \Pi(A)$ denote a specific partition of A. [3] We define $V : \Pi(A) \to \mathbb{R}$, a value function that assigns each partition $CS(A) \in \Pi(A)$ a real number. Consider an MSR in configuration $CS_{old}(A) = \{A_1^{old}, A_2^{old}, ..., A_k^{old}\}$ that reconfigures to $CS_{new}(A) = \{A_1^{new}, A_2^{new}, ..., A_{k'}^{new}\}$. Note that k and k' may be different. Such reconfigurations can happen, for example, when an MSR gets stuck at an obstacle while navigating during an exploration task. The objective of the MSR is to get into a new configuration that lets it continue performing its assigned task while incurring the minimum reconfiguration cost. We parametrize the reconfiguration cost in the following manner. Let $cost_{CS_{old}(A) \to CS_{new}(A)}(a_i, a_j)$ denote the cost that will be incurred to couple modules a_i and a_j with each other (in the process of reconfiguration from $CS_{old}(A)$ to $CS_{new}(A)$). For simplicity we will write $cost_{old,new}(a_i, a_j)$. If these two modules remain in the same MSR after reconfiguration, this cost is 0. Otherwise, this cost is given by the sum of the costs of undocking the modules a_i and a_j respectively from their current MSRs, the cost of one of the modules, say a_j, moving to the vicinity of the other module a_i and the cost of aligning and docking the two modules, as described below:[4]

$$cost_{old,new}(a_i, a_j) = \begin{cases} 0, & \text{if } a_i, a_j \in A_{k_1}^{old} \\ & \text{and } a_i, a_j \in A_{k_2}^{new} \\ cost_{Undock}(a_i) & \\ + cost_{Undock}(a_j) & \\ + cost_{Crawl}(a_j, loc(a_i)) & \\ + cost_{AlignAndDock}(a_i, a_j), & \text{otherwise} \end{cases} \quad (2)$$

Within this framework, we define the MSR self-reconfiguration problem as the following:

Definition 1. **Modular Self Reconfiguration.** Given a partition $CS_{old}(A) = \{A_1^{old}, A_2^{old}, ..., A_k^{old}\}$, find a partition $CS_{new}(A) = \{A_1^{new}, A_2^{new}, ..., A_{k'}^{new}\}$ such that the following constraint is satisfied:

$$\max_{CS_{new}(A) \in \Pi(A)} V(CS_{new}(A)) - \sum_{a_i, a_j \in A} cost_{old,new}(a_i, a_j) \quad (3)$$

4.1 Coalition Game for MSR Self-Reconfiguration

We formulate the MSR self-reconfiguration problem as finding an optimal coalition structure using a coalition game theoretic framework. Each module of the MSR is provided with a software agent that performs calculations related to the coalition game based algorithm to solve the modular

[3] For legibility, and without loss of generality, we drop the notation for time t from the MSR

[4] We assume that costs are symmetric, i.e., $cost_{old,new}(a_i, a_j) = cost_{old,new}(a_j, a_i)$.

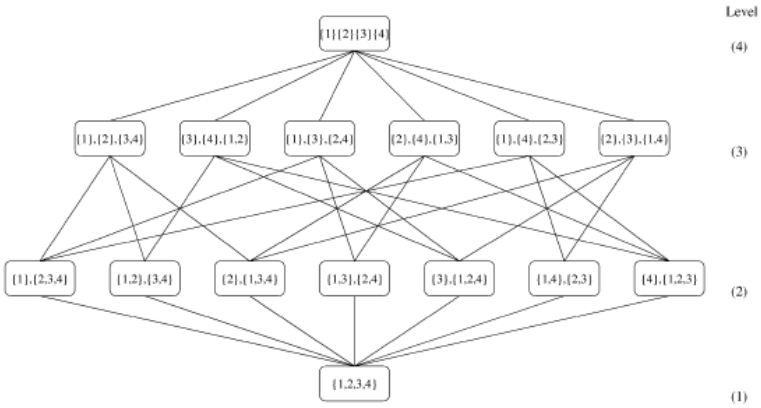


Figure 4: Coalition structure graph with 4 agents

self reconfiguration problem. We have used a popular representation of coalition games called characteristic function games (CFG) [10]. A CFG is defined by a pair of attributes (A, v), where A is the set of agents, and $v : 2^A \rightarrow \mathbb{R}$ is called a characteristic function or value function. v gives a real number called the value or worth for each possible subset or coalition S of the set of A agents. A *coalition structure* is an enumeration of the subsets S of A such that every agent appears exactly once in one of the subsets. For a set of agents A, let $\Pi(A)$ denote the set of coalition structures. For example, with $A = \{a_1, a_2, a_3\}$, $\Pi(A) = \{\{a_1\}\{a_2\}\{a_3\}, \{a_1\}\{a_2, a_3\}, \{a_2\}\{a_1, a_3\}, \{a_3\}\{a_1, a_2\}, \{a_1, a_2, a_3\}\}$. $\Pi(A)$ can be enumerated recursively as a coalition structure graph (CSG), as shown in Figure 4.1. Each coalition structure $CS(A) \in \Pi(A)$ appears as a node in the CSG. Nodes are organized into levels, and a node at level $l-1$ can be generated recursively by combining pairwise the members from disjoint partitions, for each node in level l. Each coalition structure $CS(A)$ is associated with a value $V(CS(A))$ that is usually calculated by adding the values of each coalition within the coalition structure, i.e., $V(CS(A)) = \sum_{S \in CS(A)} v(S)$. For example, with 4 agents, for the coalition structure $\{a_1, a_2\}\{a_3\}\{a_4\}$, $V(\{a_1, a_2\}\{a_3\}\{a_4\}) = v(\{a_1, a_2\}) + v(a_3) + v(a_4)$. For the context of MSR reconfiguration, an agent a_i corresponds to a single MSR-module, a coalition S corresponds to an MSR A_j, while a coalition structure corresponds to a set of MSRs. To solve the modular self-reconfiguration problem given in Definition 1, we have to find the coalition structure in the CSG that corresponds to the maximum value, i.e., find $CS^*(A) = arg \max_{CS(A) \in \Pi(a)} V(CS(A))$.

4.2 Graph-based Representation of CSG

Let $G = (A, E)$ denote a weighted, complete graph with its vertex set as the set of modules A, edge set as $E = \{(a_j, a_k) : \forall a_j, a_k \in A\}$ and edge weight function $w : E \rightarrow \mathbb{R}$ defined as:

$$w(e) = w(a_j, a_k) = Val - cost(a_j, a_k) \tag{4}$$

where Val is a fixed constant. As a simplification, we take the *cost* function to be symmetric so that $w(e) = w(e'), \forall e = (a_i, a_j), e' = (a_j, a_i) \in E$ and thus the graph can be treated as undirected. We define the sum of the edge weights in a coalition A_i to be the utility of the coalition, i.e.

$$v(A_i) = \sum_{\substack{e=(a_j,a_k): \\ a_j,a_k \in A_i, j \neq k}} w(e) \tag{5}$$

It is important to note that the edge weight function is not non-negative (which is a requirement of most graph clustering algorithms). Edges which have a positive weight correspond to a positive contribution to utility if the two modules were to join the same coalition, whereas negative edges will correspond to a decrease in utility if they were to join the same coalition. Also, with this definition, the utility of a singleton, i.e. a coalition consisting of only one module, is 0 since there are no edges within the coalition.

Graph-based MSR Reconfiguration Problem. Recall that $\Pi(A)$ is the set of all partitions of A i.e. the set of all possible non-overlapping coalition structures.

The utility of a *coalition structure* $CS(A) = \{A_1, A_2, ..., A_k\}$, is given by:

$$V(CS(A)) = \sum_{i=1}^{k} v(A_i) \tag{6}$$

Initially, A has a coalition structure consisting entirely of singletons i.e. each module is on its own and no coalitions of two or more modules have been formed. In this setting, the problem of finding an *optimal* coalition structure consists of clustering the graph $G = (A, E)$ into $CS^*(A) = \{A_1, A_2, ..., A_k\}$ such that:

$V(CS^*(A)) = \max_{CS(A) \in \Pi(A)} V(CS(A))$

4.3 Graph Clustering Approach for Coalition Formation

We will use the approach proposed in [7]. The *penalty* of a coalition structure $CS(A) = \{A_1, A_2, ..., A_k\}$ takes into account positive weighted edges *between* different coalitions in the structure and negative weighted edges *within* the same coalition in the structure and is defined to be:

$$Penalty(CS(A)) = Penalty_p(CS(A)) + Penalty_m(CS(A))$$

$$Penalty_p(CS(A)) = \sum_{\substack{e=(a_i,a_j):w(e)>0 \\ a_i \in A_{k_1}, a_j \in A_{k_2}, \\ k_1 \neq k_2}} |w(e)|$$

$$Penalty_m(CS(A)) = \sum_{\substack{e=(a_i,a_j):w(e)<0 \\ a_i,a_j \in A_{k_1}}} |w(e)|$$

The penalty of a coalition structure is equivalent to what is defined as the cost of a clustering in [7]. We use the term "penalty" so there is no confusion with the *cost* function which was defined in Equation 2. To obtain a coalition structure with close to optimal utility, we are interested in maximizing the sum of edge weights that are within coalitions in the structure. That is, it is beneficial to have modules (vertices) that have a positive edge between them to be in the same coalition, and to have modules that have a negative edge to be in different coalitions. Notice that by reducing the total weight of negative edges within coalitions, and total weight of positive edges between coalitions, the utility of the coalition is increased. By minimizing the penalty we are thus minimizing the absolute total weight of positive edges between coalitions and absolute total weight of negative edges completely within coalitions, and therefore increasing the utility of the coalition structure. In fact, as we will show, minimizing the penalty of a coalition structure is equivalent to maximizing its utility.

As in [7], for each pair of modules (vertices) i.e. for each edge $(a_i, a_j) \in E$, we introduce binary variables $x_{a_i a_j} \in \{0,1\}$ for a clustering $CS(A) = \{A_1, A_2, ..., A_k\}$ such that $x_{a_i a_j} = 0 \leftrightarrow \exists A_l \in CS(A) : a_i, a_j \in A_l$ (the two modules are in the same coalition) and $x_{a_i a_j} = 1 \leftrightarrow \exists A_{k_1}, A_{k_2} \in CS(A), k_1 \neq k_2 : a_i \in A_{k_1}, a_j \in A_{k_2}$ (the two modules are in different coalitions). Notice that $1 - x_{a_i,a_j} = 0$ iff a_i and a_j are in different coalitions and $1 - x_{a_i,a_j} = 1$ iff a_i and a_j are in the same coalition. We will use the abbreviation x_e for $x_{a_i a_j}$ where $e = (a_i, a_j) \in E$. Formulating $Penalty(CS(A))$ using these binary variables, we need non-negative constants:

$$m_e = \begin{cases} |w(e)| & \text{if } w(e) < 0 \\ 0 & \text{if } w(e) \geq 0 \end{cases}$$

$$p_e = \begin{cases} |w(e)| & \text{if } w(e) > 0 \\ 0 & \text{if } w(e) \leq 0 \end{cases}$$

So $Penalty(CS(A))$ becomes:

$$Penalty(CS(A)) = \sum_{e \in E} p_e x_e + \sum_{e \in E} m_e(1 - x_e) \qquad (7)$$

We wish to find a coalition structure with minimal cost. This is equivalent to finding the structure with optimal utility as we now show.

Proposition 1. A coalition structure with minimal penalty will have a maximal (optimal) utility.

Proof: From Equation 7 we have that the penalty of a coalition structure $CS(A)$ is:

$$Penalty(CS(A)) = \sum_{e \in E} p_e x_e + \sum_{e \in E} m_e(1 - x_e)$$

Using the same notation, we have that the utility of a coalition structure from Equations 5, 6 is:

$$V(CS(A)) = \sum_{i=1}^{k} v(A_i) = \sum_{i=1}^{k} \sum_{\substack{e=(a_j,a_k): \\ a_j,a_k \in A_i, j \neq k}} w(e) =$$
$$\sum_{e \in E} p_e(1 - x_e) - \sum_{e \in E} m_e(1 - x_e) =$$
$$\sum_{e \in E} p_e - \left(\sum_{e \in E} p_e x_e + \sum_{e \in E} m_e(1 - x_e) \right) =$$
$$\sum_{e \in E} p_e - Penalty(CS(A))$$

Since $\sum_{e \in E} p_e$ is a constant (the variables are the x_e's), then minimizing $Penalty(CS(A))$ is equivalent to maximizing $V(CS(A))$ and thus the coalition with minimal penalty will have the maximal (optimal) utility.

□

Notice that $Penalty(CS(A)) \geq 0, \forall CS(A) \in \Pi(A)$ and to minimize the Penalty using the above formulation, it would suffice to set $x_e = 0$ whenever $p_e > 0$ and $x_e = 1$ whenever $m_e > 0$ so that $Penalty = 0$, however this will not necessarily correspond to a valid coalition structure since the x_e variables are not independent. As noted in [7], an assignment of values $\{0,1\}$ to the variables x_e corresponds to a valid coalition structure (clustering) if the variables satisfy the triangle inequality, that is $\forall a_i, a_j, a_k \in A, i \neq j \neq k$, $x_{a_i,a_j} + x_{a_j,a_k} \geq x_{a_i,a_k}$. The problem can therefore be formulated as a 0-1 integer linear program:

$$\text{min: } \sum_{e \in E} p_e x_e + \sum_{e \in E} m_e(1 - x_e)$$

constraints:
$x_{a_i,a_j} \in \{0,1\}, \forall a_i, a_j \in A, i \neq j$
$x_{a_i,a_j} + x_{a_j,a_k} \geq x_{a_i,a_k}, \forall a_i, a_j, a_k \in A, i \neq j \neq k$
$x_{a_i,a_j} = x_{a_j,a_i} \forall a_i, a_j \in A, i \neq j$

As is known, 0-1 integer linear programming is NP-complete, so the problem is relaxed to a linear program [7]:

$$\text{min: } \sum_{e \in E} p_e x_e + \sum_{e \in E} m_e(1 - x_e)$$

constraints:
$x_{a_i,a_j} \in [0,1], \forall a_i, a_j \in A, i \neq j$
$x_{a_i,a_j} + x_{a_j,a_k} \geq x_{a_i,a_k}, \forall a_i, a_j, a_k \in A, i \neq j \neq k$
$x_{a_i,a_j} = x_{a_j,a_i} \forall a_i, a_j \in A, i \neq j$

The problem can now be solved in polynomial time, though it may yield a fractional solution i.e. the variables x_e are in the interval $[0,1]$, and are not necessarily binary. In this case the authors in [7] propose a rounding algorithm based on region growing to obtain a valid approximate solution i.e. obtain a valid assignment of 0's and 1's to the x_e variables which will yield a close to minimal penalty. The whole algorithm runs in polynomial time (polynomial in the number of variables) and is a $O(\log n)$ approximation. The number of variables is simply the number of edges in

the graph. In our case, we have a complete graph so that $|E| = \binom{|A|}{2} = \binom{n}{2} = \frac{n(n-1)}{2}$ and so the algorithm will run in time polynomial in the number of modules (vertices) n. The algorithm is outlined as follows:

1. Coordinates of all agents in A are specified. Parameter Val is specified.

2. $\forall a_i, a_j \in A$, $w(a_i, a_j)$ is calculated using Equations 2 and 4.

3. Objective function given by Equation 7 is formulated and constraints are set.

4. Linear programming is used to obtain a solution to the optimization problem.

5. If the solution is 0-1 integer then a valid coalition structure has been found. Else if the solution is fractional, the region growing rounding algorithm [7] is used to obtain a valid approximate solution.

5. EXPERIMENTAL RESULTS

We have implemented the described algorithm for obtaining optimal coalition structures using the mixed integer linear programming solver LPsolve version 5.5.2.0. In this section we present results on simulations for agent set sizes from 3 to 43. For agent set sizes between 3 and 12, we were able to perform an exhaustive search on the space of all coalition structures to find the actual optimal coalition structure and compare it with the structure obtained using the graph clustering linear programming model. For higher agent set sizes the exhaustive search (complexity $O(n^n)$) becomes prohibitive, as do the algorithms for the general CSG problem [11, 15]. The reason we are able to outperform those methods is that our formulation is a restricted case i.e. we are restricting the problem to a weighted graph where coalition utilities are calculated by summing pairwise utilities (edge weights).

For each of the agent set sizes $n = \{3, 4, ..., 43\}$, we used a grid size of $(n+4) \times (n+4)$ and generated random integer coordinates for each of the agents i.e. for each agent $a_i \in A, 1 \leq i \leq n$, we assigned coordinates (x_{a_i}, y_{a_i}) where x_{a_i} is randomly generated from $\{0, 1, ..., n+3\}$ and y_{a_i} is randomly generated from $\{0, 1, ..., n+3\}$. We generated 30 random arrangements of n agents in an $(n+4) \times (n+4)$ grid for each value of n from 3 to 12, and 10 random arrangements for n from 13 to 43. The edge weights were calculated using $w(e) = Val - cost(a_j, a_k)$, and for simplification we took $cost(a_j, a_k) = d(a_j, a_k) + cost_{AlignAndDock}$ where $d(a_j, a_k)$ is the Euclidean distance between agents a_j, a_k in the grid and $cost_{AlignAndDock} = 1$ was a constant representing the cost of alignment and docking of two modules (agents). We set $Val = \frac{n}{2} + 3$ for each different value of n. Val was chosen so that for each edge (a_i, a_j), the weight is $w(e) = Val - cost(a_j, a_k) = \frac{n}{2} + 3 - d(a_j, a_k) - 1 = \frac{n+4}{2} - d(a_j, a_k)$ meaning if modules a_i, a_j are within a distance of $\frac{n+4}{2}$ (half the width of the grid) then their edge weight (utility) will be positive, and if they are at least half a grid width apart then they will have a negative edge weight. For singleton coalitions we used the utility of 0. Notice that in the graph formulation, singleton coalitions have no edges contained and hence do not contribute anything to the coalition utility[5]. A zero utility for singletons also indicates that a coalition consisting of a single module will not provide any contribution to the total utility of the coalition structure.

Such a scenario arises in a practical setting where the modules are deployed in an unknown environment as singletons. Deploying a configured MSR is harder than deploying single modules separately. Each of the individual modules may be dropped from an aircraft/spacecraft with a parachute and thus will be scattered in the ground environment once they land. Our goal is then to find the optimal coalition structure so that the modules can form into larger MSRs and proceed with exploration.

After randomly generating the test cases for each agent set size and calculating the edge weights, we formulated the objective function (the penalty of a coalition structure) to minimize and the constraints, then used LPsolve to produce a valid solution. For agent set sizes up to 43, using the default settings in LPsolve, all solutions to the randomly generated test cases were 0-1 integer i.e. no rounding is required. In this case an exact solution is obtained to the 0-1 integer linear programming model and thus it is an optimal coalition structure. As empirical evidence of this we took the actual optimal utility from the exhaustive search and compared to the value obtained using LPsolve.

Table 1 shows the mean ratio (averaged over the 30 test cases for each agent set size) of the utility of the coalition structure we found to the utility of the optimal coalition structure. Mean runtime (average over each of the 30 test cases) is also displayed. For implementation we used a desktop PC (Intel Core i7 - 960 3.20GHz, 12GB DDR3 SDRAM)

No. of Agents	Mean Ratio to Optimal Utility	Mean Runtime (secs)
3	1	0.004667
4	1	0.004733
5	1	0.004833
6	1	0.005
7	1	0.005233
8	1	0.005733
9	1	0.006367
10	1	0.007233
11	1	0.008567
12	1	0.01127

Table 1. Mean Ratio of Utilities for Coalition Structures Obtained to Optimal Coalition Structures

From Table 1, we see that the method we implemented is able to find optimal coalition structures for the given number of agents. In fact, in all of the 30 cases for each agent size, optimal coalition structures were found.

Figure 5 shows running times averaged over 10 test cases for each value of agent set size from 3 to 43. The algorithm was implemented for agents set sizes 13 to 43 on the same machine. Notice that even for 43 agents, the running time is approximately 5 seconds. Running algorithms which explore the space of all coalition structures for such numbers

[5]While realistically, singleton coalitions should have a value, we do not consider this in our formulation. Adding vertex weights to the formulation is a possible way to account for singleton coalitions.

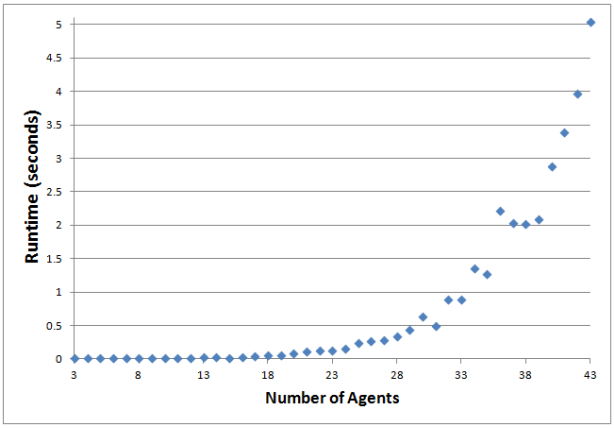


Figure 5: Average running times for various agent set sizes

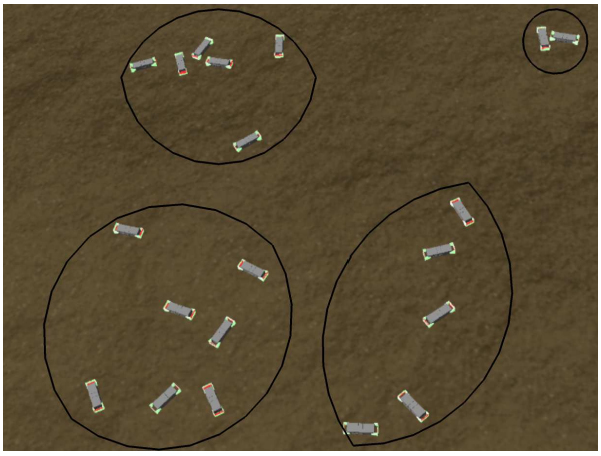


Figure 6: Initial locations of 20 ModRED modules randomly scattered within an environment (snapshot from Webots)

of agents is prohibitive. The algorithms are all bounded below by $\Omega(2^n)$ since the utility of every possible subset of A has to be obtained. While the algorithms [15], [11] are applicable to any coalition game i.e. any valuation function, the method implemented by us is only applicable to the graph coalition game case where the valuation function is calculated by summing pairwise utilities.

Figure 6 shows the initial random arrangement of 20 ModRED modules in a 24×24 grid (snapshot taken from Webots). Each module is assumed to be in its own singleton coalition, edge weights are calculated using equation 5 with $Val = 15$, $cost = 1$, and Euclidean distances between modules. The coalition structure obtained using the graph clustering method with the given parameters is displayed (modules in the same coalition are circled). The clustering method naturally groups close modules into the same cluster (since with constant *cost*, edge weights are primarily influenced by distance). Once the coalition structure is determined, the modules are instructed to form into their respective coalitions and configure into chains as is illustrated in Figure 7.

For agent set sizes larger than 43, LPsolve produced fractional solutions i.e. the edge variables x_{a_i,a_j} were values in the interval (0,1). In this case, the region growing rounding algorithm in [7] is implemented to obtain $O(\log n)$-approximations to the optimal coalition structure. Additional time is required for this procedure but it should be noted that it runs in polynomial time and therefore does not change the fact that the overall algorithm runs in polynomial time.

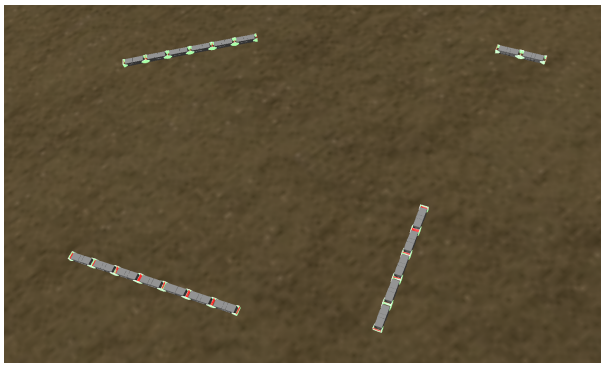


Figure 7: ModRED modules form the specified coalitions and configure into four different chain formations (snapshot from Webots)

6. CONCLUSION AND FUTURE WORK

We have formulated the CSG problem for our setting of MSR reconfiguration as a graph coalition formation game. Current state of the art algorithms for general coalition formation are all bounded below by $\Omega(2^n)$ [15], [11] and for a guaranteed optimal solution, the worst case running time is $O(n^n)$. In our formulation, although it is a specific case with a graph representation, there is no longer a $\Omega(2^n)$ lower bound and algorithms exist [7] which guarantee a $O(\log n)$ approximation and run in polynomial time. In this setting, coalition formation for large sets of agents becomes feasible.

In general, solving the 0-1 integer linear programming problem (which is part of the graph clustering technique) is NP-Complete, but when the problem is relaxed to a general linear programming problem, solutions can be found in polynomial time. Fractional solutions do not represent a valid coalition structure and a $O(\log n)$ approximation polynomial time algorithm is used to obtain a valid coalition structure. In the simulation results presented, we did not have to implement the region growing rounding algorithm in [7] since we obtained 0-1 integer solutions and thus optimal coalition structures. As the agent set size grows larger, the solutions obtained in the linear programming part of the algorithm are fractional and region growing has to be applied to obtain valid coalition structures. We plan to extend our work to include the rounding algorithm so that the problem can be approximately solved for larger agent set sizes and compare it with more recent coalition structure search algorithms [12].

Also, our current formulation is able to produce a close to optimal coalition structure when reconfiguring from the coalition structure in which each module is in a coalition on its own i.e. each coalition is a singleton. We plan to extend the approach so that we can reconfigure from any given initial configuration. Assigning tasks for the coalitions is another extension we hope to explore. In this scenario

each coalition is assigned a task and constraints are set so that each coalition is able to solve the assigned task. In the case when not all tasks can be solved, reconfiguration has to be performed to obtain a coalition structure which will meet the task needs. Implementing our approach in the physical world to the MSR ModRED requires that we take into account uncertainty since sensor noise becomes a key issue. Extending our formulation by using stochastic edge weights is one approach to tackling the issue of uncertainty in the graph coalition formation problem.

7. REFERENCES

[1] Y. Bachrach, P. Kohli, V. Kolmogorov, and M. Zadimoghaddam. Optimal coalition structures in graph games. *arXiv:1108.5248v1 [cs.GT]*, 2011.

[2] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[3] Z. Butler, S. Brynes, and D. Rus. Distributed motion planning for modular robots with unit decompressable modules. In *IEEE/RSJ Intl. Conf. Intell. Rob. and Sys.*, pages 790–796, Maui, Hawaii, 2001.

[4] A. Castano, W. Shen, and P. Will. Conro: Towards deployable robots with inter-robots metamorphic capabilities. *Autonomous Robots*, 8:309–324, 2000.

[5] G. Chirikjian, A. Pamecha, and I. Ebert-Upfhoff. Evaluating efficiency of self reconfiguration in a class of modular robots. *Robotics Systems*, 13:317–338, 1996.

[6] K. Chu, S. G. M. Hossain, and C. Nelson. Design of a four-dof modular self-reconfigurable robot with novel gaits. In *ASME International Design Engineering Technical Conference*, pages DETC2011–47746, Washington, D.C., 2011.

[7] E. Demaine and N. Immorlica. Correlation clustering with partial information. *Lecture Notes in Computer Science*, 2764:71–80, 2003.

[8] X. Deng and C. Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics and Operations Research*, 19(2):257–266, 1994.

[9] A. Kamimura, E. Yoshida, S. Murata, H. Kurokawa, K. Tomita, and S. Kokaji. Distributed self-reconfiguration of m-tran iii modular robotic system. *Intl. J. of Rob.*, 27(3-4):373–386, 2008.

[10] R. Myerson. *Game Theory: Analysis of Conflict*. Cambridge, Massachusetts: Harvard University Press, 1997.

[11] T. Rahwan. *Algorithms for Coalition Formation in Multi-Agent Systems*. PhD thesis, University of Southampton, 2007.

[12] T. Rahwan, S. Ramchurn, A. Giovannucci, and N. Jennings. An anytime algorithm for optimal coalition structure generation. *Journal of Artificial Intelligence Research*, 34:521–567, 2009.

[13] D. Ray. *A Game-Theoretic Perspective on Coalition Formation (1st ed.)*. Oxford University Press, USA, 2008.

[14] M. Rosa, S. Goldstein, P. Lee, J. Campbell, and P. Pillai. Scalable shape sculpturing via hole motions. In *IEEE Intl. Conf. Rob. and Auton.*, pages 1462–1468, Orlando, FL, 2006.

[15] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2):209–238, 1999.

[16] K. Stoy, D. Brandt, and D. Christensen. *Self-Reconfigurable Robots: An Introduction*. Cambridge, Massachusetts: The MIT Press, 2010.

[17] T. Voice, M. Poulakarov, and N. Jennings. Graph coalition structure generation. *arXiv:1102.1747v1 [cs.DS]*, 2011.

[18] M. Yim and et al. Modular self-reconfigurable robot systems: Challenges and opportunities for the future. *IEEE Robotics and Automation Magazine*, 14(1):43–53, 2007.

UT Austin Villa 2011: A Champion Agent in the RoboCup 3D Soccer Simulation Competition

Patrick MacAlpine, Daniel Urieli, Samuel Barrett, Shivaram Kalyanakrishnan[*],
Francisco Barrera, Adrian Lopez-Mobilia, Nicolae Ştiurcă[†], Victor Vu, and Peter Stone
Department of Computer Science, The University of Texas at Austin, Austin, TX 78701, USA
{patmac, urieli, sbarrett, shivaram,
tank225, alomo01, nstiurca, diragjie, pstone}@cs.utexas.edu

ABSTRACT

This paper presents the architecture and key components of a simulated humanoid robot soccer team, UT Austin Villa, which was designed to compete in the RoboCup 3D simulation competition. These key components include (1) an omnidirectional walk engine and associated walk parameter optimization framework, (2) an inverse kinematics based kicking architecture, and (3) a dynamic role assignment and positioning system. UT Austin Villa won the RoboCup 2011 3D simulation competition in convincing fashion by winning all 24 games it played. During the course of the competition the team scored 136 goals while conceding none. We analyze the effect of each component in isolation and show through extensive experiments that the complete team significantly outperforms all the other teams from the competition.

Categories and Subject Descriptors

I.2.9 [**Computing Methodologies**]: Artificial Intelligence—*Robotics*

General Terms

Algorithms, Design, Experimentation

Keywords

Humanoid robotics, Robot soccer, Machine learning

1. INTRODUCTION

Robot Soccer [3] has served as an excellent research domain for autonomous agents and multi-agent systems over the past decade and a half. In this domain, teams of autonomous robots compete with each other in a complex, real-time, noisy and dynamic environment, in a setting that is both collaborative and adversarial. Robot soccer has spread over several popular platforms, each having its own advantages. For example, the real robot competitions, including the humanoid robot league, have typically emphasized low-level robot control challenges. On the other hand, the RoboCup 2D simulation platform has emphasized high-level team strategy challenges. In this paper, we focus on the RoboCup 3D simulation platform, which integrates both these low-level and high-level challenges under one umbrella.

In the 3D simulation league teams of nine simulated humanoids play in a simulation environment with realistic physics, state-noise, multidimensional actions and real-time control. One advantage of the 3D simulation domain over real robots is avoiding the high cost of errors, and the relatively slow feedback loop, that happens when testing new skills in the real world. An advantage over the 2D simulator is the ability to test high-level team strategies under the constraints of humanoid locomotion. Due to the complexity of the environment, parts of the agent are hard to design by hand. For instance, it is a significant challenge to design a walk that is both fast and stable. The 3D simulation platform allows for designing and investigating general methodologies for skill and strategy acquisition in a complex, challenging domain, using machine learning.

In this paper, we present UT Austin Villa, the winning agent of the 3D simulation league in RoboCup 2011. Each of UT Austin Villa's field players is controlled by (a separate instance of) the same program. The players continually estimate the world state from noisy observations, reason about position assignments, and then quickly and robustly move on the field using a learned walk. In this paper, we describe the complete agent, but focus particularly on the most novel components that were key contributors to our success. Specifically, we focus on (1) an omnidirectional walk agent and an associated walk parameter optimization framework, (2) an automatically optimized inverse kinematics based kicking architecture, and (3) a dynamic role assignment and positioning system. We analyze the individual components and the complete team's performance both in competition and in controlled experiments.[1]

The rest of the paper is structured as follows. Section 2 gives a domain description. Section 3 describes our agent's architecture. Section 4, 5 and 6 describe the three key components of our agent, respectively. Results are given in Section 7, and Section 8 summarizes.

2. DOMAIN DESCRIPTION

Robot soccer has served as an excellent platform for testing learning scenarios in which multiple skills, decisions, and

[*]S. Kalyanakrishnan is currently at Yahoo! Labs.

[†]N. Ştiurcă is currently at the University of Pennsylvania.

[1]Videos of these components in action can be found online at `http://www.cs.utexas.edu/~AustinVilla/sim/3dsimulation/AustinVilla3DSimulationFiles/2011/html/components.html`

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

controls have to be learned by a single agent, and agents themselves have to cooperate or compete. There is a rich literature based on this domain addressing a wide spectrum of topics from low-level concerns, such as perception and motor control [6, 12], to high-level decision-making problems [10, 13].

The RoboCup 3D simulation environment is based on SimSpark [4], a generic physical multiagent system simulator. SimSpark uses the Open Dynamics Engine [2] (ODE) library for its realistic simulation of rigid body dynamics with collision detection and friction. ODE also provides support for the modeling of advanced motorized hinge joints used in the humanoid agents.

The robot agents in the simulation are homogeneous and are modeled after the Aldebaran Nao robot [1], which has a height of about 57 cm, and a mass of 4.5 kg. The agents interact with the simulator by sending torque commands and receiving perceptual information. Each robot has 22 degrees of freedom: six in each leg, four in each arm, and two in the neck. In order to monitor and control its hinge joints, an agent is equipped with joint perceptors and effectors. Joint perceptors provide the agent with noise-free angular measurements every simulation cycle (20 ms), while joint effectors allow the agent to specify the torque and direction in which to move a joint. Although there is no intentional noise in actuation, there is slight actuation noise that results from approximations in the physics engine and the need to constrain computations to be performed in real-time. Visual information about the environment is given to an agent every third simulation cycle (60 ms) through noisy measurements of the distance and angle to objects within a restricted vision cone (120°). Agents are also outfitted with noisy accelerometer and gyroscope perceptors, as well as force resistance perceptors on the sole of each foot. Additionally, agents can communicate with each other every other simulation cycle (40 ms) by sending messages limited to 20 bytes. Figure 1 shows a visualization of the Nao robot and the soccer field during a game.

3. AGENT ARCHITECTURE

The UT Austin Villa agent receives visual sensory information from the environment which provides distances and angles to different objects on the field. It is relatively straightforward to build a world model by converting this information about the objects into Cartesian coordinates. This of course requires the robot to be able to localize itself for which the agent uses a particle filter. In addition to the vision perceptor, the agent also uses its accelerometer readings to determine if it has fallen and employs its auditory channels for communication.

Figure 1: A screenshot of the Nao humanoid robot (left), and a view of the soccer field during a 9 versus 9 game (right).

Once a world model is built, the agent's control module is invoked. At the lowest level, the humanoid is controlled by specifying torques to each of its joints. This is implemented through PID controllers for each joint, which take as input the desired angle of the joint and compute the appropriate torque. Further, the agent uses routines describing inverse kinematics for the arms and legs. Given a target position and pose for the hand or the foot, the inverse kinematics routine uses trigonometry to calculate the target angles for the different joints along the arm or the leg to achieve the specified target, if possible.

The PID control and inverse kinematics routines are used as primitives to describe the agent's skills. In order to determine the appropriate joint angle sequences for walking and turning, the agent utilizes an omnidirectional walk engine which is described in Section 4. When invoking the kicking skill, the agent uses inverse kinematics to control the trajectory of the kicking foot as discussed in Section 5. Two other useful skills for the robot are falling (for instance, by the goalie to block a ball) and rising from a fallen position.

It is worth mentioning that some of the agent's skills, like diving, rising from a fall, and kicking, are defined using a flexible text-file-based skill description language, which was used by our team in RoboCup 2010 [15], and which allows to quickly create new skills, while leaving some of the skills' parameters opened for optimization using machine learning.

Because the team's emphasis was mainly on learning robust and stable low-level skills, the high-level team strategy is relatively straightforward. The player closest to the ball is instructed to go to it while other field player agents dynamically choose target positions on the field as explained in Section 6. The goalie is instructed to stand a little in front of its goal and, using a Kalman filter to track the ball, attempts to dive and stop the ball if it comes near.

4. OMNIDIRECTIONAL WALK ENGINE AND OPTIMIZATION

The primary key to UT Austin Villa's success in the 2011 RoboCup 3D simulation competition was its development and optimization of a stable and robust fully omnidirectional walk. The team used an omnidirectional walk engine based on the research performed by Graf et al. [8]. The main advantage of an omnidirectional walk is that it allows the robot to request continuous velocities in the forward, side, and turn directions, permitting it to approach its destination more quickly. In addition, the robustness of this engine allowed the robots to quickly change directions, adapting to the changing situations encountered during soccer games.

4.1 Walk Engine Implementation

The walk engine uses a simple set of sinusoidal functions to create the motions of the limbs with limited feedback control. It processes desired walk velocities given as input, chooses destinations for the feet and torso, and then inverse kinematics are used to determine the joint positions required. Finally, PID controllers for each joint convert these positions into torque commands that are sent to the joints.

The walk engine first selects a trajectory for the torso to follow, and then determines where the feet should be with

respect to the torso location. The trajectory is chosen using a double linear inverted pendulum, where the center of mass is swinging over the stance foot. In addition, as in Graf et al.'s work [8], the simplifying assumption that there is no double support phase is used, so that the velocities and positions of the center of mass must match when switching between the inverted pendulums formed by the respective stance feet. Further details of the walk can be found in [11].

The walk engine is parameterized using more than 40 parameters, ranging from intuitive quantities, like the step size and height, to less intuitive quantities like the maximum acceptable center of mass error. These parameters are initialized based on an understanding of the system and also testing them out on an actual Nao robot. This initialization resulted in a stable walk. However, the walk was extremely slow compared to speeds required during a competition. We refer to the agent that uses this walk as the *Initial* agent.

4.2 Walk Engine Parameter Optimization

The slow speed of the *Initial* agent calls for using machine learning to obtain better walk parameter values. Parameters are optimized using the CMA-ES algorithm [9], which has been successfully applied in [15]. CMA-ES is a policy search algorithm that successively generates and evaluates sets of candidates. Once CMA-ES generates a group of candidates, each candidate is evaluated with respect to a *fitness* measure. When all the candidates in the group are evaluated, the next set of candidates is generated by sampling with probability that is biased towards directions of previously successful search steps.

As optimizing 40 real-valued parameters, can be impractical, a carefully chosen subset of 14 parameters was selected for optimization while keeping all the other parameters fixed. The chosen parameters are those that have the highest potential impact on the speed and stability of the robot, and are mainly: rotation and height; the robot's center of mass height, shift amount, and default position; the fraction of time a leg is on the ground and the time allocated for one step phase; the step size PID controller; center of mass normal error and maximum acceptable errors; and the robot's forward offset.

Similarly to a conclusion from [15], we have found that optimization works better when the robot's fitness measure is its performance on tasks that are *executed during a real game*. This stands in contrast to evaluating it on a general task such as the speed of walking straight. Therefore, the robot's in-game behavior is broken down into a set of smaller tasks, and the parameters for each one of these tasks is sequentially optimized. When optimizing for a specific task, the performance of the robot on the task is used as CMA-ES's fitness value for the current candidate parameter set values.

In order to simulate common situations encountered in gameplay, the walk engine parameters are optimized for a goToTarget subtask. This consists of an obstacle course in which the agent tries to navigate to a variety of target positions on the field. The goToTarget optimization[2] includes quick changes of target/direction for focusing on the reaction speed of the agent as well as holding targets for longer durations to improve the straight line speed of the agent. Additionally the agent is instructed to stop at different times during the optimization to ensure that it is stable and does not fall over when doing so. In order to encourage the agent to learn both quick turning behavior and a fast forward walk, the agent always walks and turns toward its designated target at the same time. This allows for the agent to swiftly adjust and switch its orientation to face its target, thereby emphasizing the amount of time during the optimization that it is walking forward. Optimizing the walk engine parameters in this way resulted in a significant improvement in performance with the *GoToTarget* agent able to quickly turn and walk in any direction without falling over. This improvement also showed itself in actual game performance as when the *GoToTarget* agent played 100 games against the *Initial* agent, the *GoToTarget* agent won on average by 8.82 goals with a standard error of .11.

To further improve the forward speed of the agent, a second walk engine parameter set is optimized for walking straight forward. This is accomplished by running the goToTarget subtask optimization again, but this time the *goToTarget* parameter set is held fixed while a new parameter set, called the *sprint* parameter set, is learned. The *sprint* parameter set is used when the agent's orientation is within 15° of its target. By learning the *sprint* parameter set in conjunction with the *goToTarget* parameter set, the new *Sprint* agent remains stable while switching between the two walk parameter sets, and the agent's speed increases from .64 m/s to .71 m/s as timed when walking forward for ten seconds after starting from a stand still.

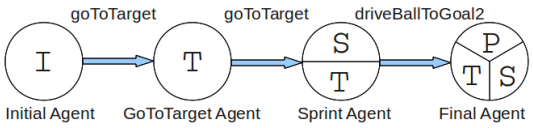


Figure 2: UT Austin Villa's walk parameter optimization progression. Circles represent the set(s) of parameters used by each agent during the optimization progression while the arrows and associated labels above them indicate the optimization tasks used in learning. Parameter sets are the following: I = *initial*, T = *goToTarget*, S = *sprint*, P = *positioning*.

In the next step we further optimize the agent to quickly position near the ball. While the goToTarget optimization emphasizes quick turns and forward walking speed, positioning around the ball involves more side-stepping to circle the ball. To account for this discrepancy, the agent learns a third parameter set called the *positioning* parameter set. To learn this new parameter set a driveBallToGoal2[3] optimization task is created, in which the agent is evaluated on how far it is able to dribble the ball over 15 seconds when starting from a variety of positions and orientations from the ball. Whenever the agent enters a radius of .8 meters from the ball, it transitions to using the *positioning* parameter set. During the optimization, both the *goToTarget* and *sprint* parameter sets are held fixed. As the optimization naturally includes

[2]Note that we use three types of notation for each of goToTarget, *GoToTarget*, *goToTarget*, to distinguish between an optimization task, an agent created by this optimization task and a parameter set. Similarly for "sprint" and "initial".

[3]The '2' at the end of the name driveBallToGoal2 is used to differentiate it from a driveBallToGoal optimization that was used in [15].

transitions between all three parameter sets, this constrains all parameter sets to be compatible with each other. Adding both the *positioning* and *sprint* parameter sets further improves the agent's performance such that the resulting *Final* agent, is able to beat the *GoToTarget* agent by an average of .24 goals with a standard error of .08 across 100 games. A summary of the progression in optimizing the three different walk parameter sets can be seen in Figure 2.

5. KICK ENGINE

While the learned walk described in Section 4 is by far the aspect of UT Austin Villa that is most responsible for its success, as is affirmed in Section 7, robust and accurate kicking is another skill that is essential for playing soccer at a high level.

To motivate some of the design decisions in our kick engine which we discuss in depth later in this section, we first present the desired qualities of the engine. For a kick to be broadly applicable, it needs to be agile, robust, versatile, and easily and concisely parameterizable. *Agility* refers to taking shots quickly. *Robustness* entails taking accurate and powerful shots in spite of positioning errors (e.g., without the agent being perfectly lined up with the ball). *Versatility* refers to being able to kick in multiple directions from multiple ball starting locations. The parameterization criterion serves to facilitate learning optimized kicks.

5.1 Kick Engine Implementation

To achieve these criteria, our kick engine employs a system of defining and dynamically computing smooth curves which guide the foot's trajectory through the ball at high speed and in the desired direction. We use Cubic Hermite Splines to define the foot trajectories. Agility and robustness are achieved by defining the kick trajectory relative to the ball in Cartesian space. Unlike our previous year's team which used fixed joint angle skills exclusively, the current agents do not have to tip-toe eg., directly behind the ball at a set distance in order to kick the ball eg., forward. Instead, the kick engine dynamically computes the trajectory of the foot once the agent is close enough to the ball, regardless of whether the agent finished positioning or whether the agent was able to position itself precisely relative to the ball. Versatility is achieved because multiple directional kicks can be defined and used at will. Learning and optimization of kicks is facilitated by the parameterization of the foot trajectories in terms of a sparse set of control (way-) points. The flow of the kick engine follows.

First, a kick is selected, and the agent approaches the ball (Section 5.1.1). Once close enough to the ball, it shifts its weight onto the support foot and computes the kicking foot trajectory necessary to perform the desired kick (Section 5.1.2). At each time step during the kick, the kick engine interpolates the control (way-) points defined in the kick skill file (Section 5.1.5) to produce a target pose for the foot in Cartesian space (Section 5.1.3). Finally, an IK solver computes the necessary joint angles of the kicking leg, and these angles are fed to the joint PID controllers (Section 5.1.4). Figure 3 illustrates the program flow of the kick engine.

5.1.1 Kick Choice and Ball Approach

As the agent approaches the ball, it must decide which type of kick to attempt (Section 5.1.6 describes the options)

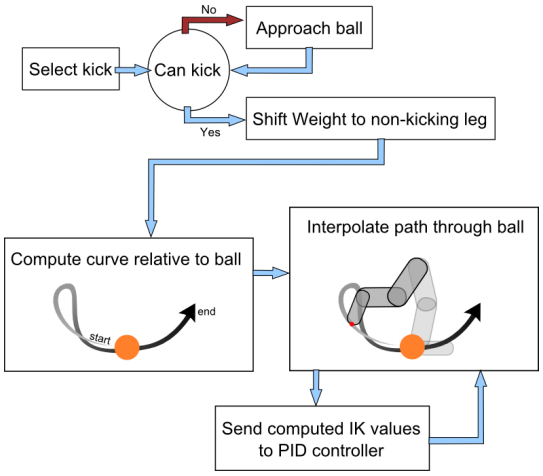


Figure 3: The flow of the agent deciding when to kick the ball and how to interpolate the curve created relative to the ball.

and whether to use the left or right foot. Each kick skill definition includes a target offset of the agent relative to the ball. Choosing a kick reduces to choosing the target with the lowest cost for the agent to move to. We calculate the cost of each target through the following variables and formula:

$$
\begin{aligned}
distCost &= |\text{agentPosition} - \text{targetOffsetPosition}|\,/m \\
turnCost &= \frac{|\text{agentOrientation} - \text{targetOrientation}|}{360^\circ} \\
ballPenalty &= \begin{cases} .5 & \text{if ball is in path to target offset} \\ 0 & \text{otherwise} \end{cases} \\
kickCost &= \text{distCost} + \text{turnCost} + \text{ballPenalty}
\end{aligned}
$$

The chosen target is approached using the walk engine. During approach, the kick engine continuously checks if the agent is close enough to kick by using the IK solver to determine if the foot can reach most ($> 90\%$) of the points along the trajectory for the chosen kick.

5.1.2 Dynamically Compute Kick Trajectory

Once the agent has shifted its weight in preparation for a kick, it notes the ball's position with respect to itself (specifically its torso, the root of the leg kinematic chains). This offset is added to the control points in the kick skill file to dynamically compute the exact curve of the foot with respect to the agent's torso.

5.1.3 Interpolate Kick Trajectory

The control points defined in the kick skill files are used to compute a smooth 3D curve. We use the Cubic Hermite Spline formulation to interpolate the control points because Hermite Splines yield curves with C^1 continuity which pass through all control points [5]. The time offset from the start of the kick is normalized to the range $[0-1]$ (0 is the start of the kick; 1 is the end), and the normalized offset is used to sample the Hermite Spline. The kick skill files also define the Euler angles (roll, pitch, and yaw) of the foot at each control point. These angles are linearly interpolated.

5.1.4 Kick Inverse Kinematics

For the inverse kinematics calculations, we used Open-

RAVE's [7] analytic inverse kinematics solver. The OpenRAVE IK solver can process arbitrary forward kinematic chains defined in XML and produce fast C++ source code that solves the inverse kinematics. Note that the time-consuming analytic processing is done offline, and the fast C++ code can be queried hundreds of times at each time step without a significant computational cost.

5.1.5 Kick Skill Definition

Extending the skill definition files to allow Cartesian coordinate plus Euler angle waypoints for each foot, we predefine all six degree of freedom positions of the foot for a given curve at any linear time through the curve.

5.1.6 Directional Kicks

We defined five kicks that assume that the ball is in front of the agent such that it can kick directly forward and at 45° and 90° angles either outward or inward, depending on which leg is used. We also created directional kicks which assume that the ball is to the side of or behind one of the legs. See Figure 4.

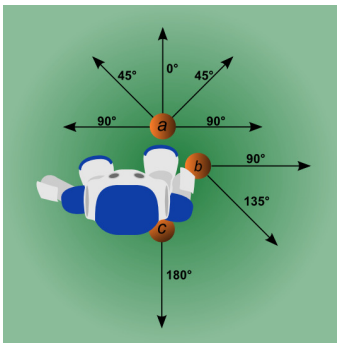


Figure 4: The agent can dynamically kick the ball in varied directions with respect to the placement of the ball at *a*, *b*, and *c*.

5.2 Kick Optimization

We can then optimize the waypoints (three to five per kick) for kicked distance and speed through CMA-ES. This then allows us to have multiple directional kicks defined through simple curves as we do not have to dedicate large amounts of time tweaking each one and can create rough paths to guide the initial seed of the agent's kick.

In order to learn the parameters for a kick we set up an optimization task where the agent approaches the ball from ten different angles along a half circle arc around the ball and attempts to kick the ball toward a specific target. The parameters being optimized are the XYZ and RPY values of the waypoints that define the curve of the kick, how quickly the kicking foot moves through the curve, and also the target offset from the ball to move toward during the kick approach. The fitness of an agent is measured by the average distance the ball travels toward the target across all kick attempts. The agent is given a penalty fitness of -1 for every kick during which it falls over, runs into the ball, or isn't able to kick the ball after ten seconds have passed. Penalizing the agent for taking too long to kick encourages kicking agility while having the agent approach the ball from multiple angles and penalizing for falling promote kicking robustness.

5.3 Kick Performance

While our kicking system shows a lot of promise, we found out after the competition that our agent does slightly better without kicking turned on during self play. A version of our agent with the kicking system turned off was able to beat our agent that does kick by an average of .15 goals per game across 100 games with a standard error of .07. This resulted in a tally of 27 wins for the agent that does not kick, 12 wins for that agent that does kick, and 61 ties. We believe the reason for this slight degradation in performance when kicking is due to our kicking agent needing to slow down a little when approaching the ball to kick it, instead of maintaining a full speed walk while dribbling the ball, so as to not accidentally run into the ball. Additionally we have yet to implement a strategy for passing and only kick in the direction we want to dribble if an opponent agent is approaching to take the ball away. We therefore include a description of the kick in this paper as a key component of the overall agent, even though it was not necessary for winning this year's competition.

With better tuning such that the agent can approach the ball without needing to slow down, and the addition of a strategy to take full advantage of the ability for kicking to quickly move the ball, we expect our kick system to provide a substantial gain in the performance of the agent. The kicking system has already shown some promise when used with walks that are not as effective at dribbling as our current walk. When playing kicking and non-kicking versions of our agent with slow *initial* walk parameters, as described in Section 4.1, against each other the kicking agent scored 8 goals while the non-kicking agent failed to score.

6. DYNAMIC ROLE ASSIGNMENT AND POSITIONING SYSTEM

While low level skills such as walking and kicking are vitally important for having a successful soccer playing agent, the agents must work together as a team in order to maximize their game performance. One often thinks of the soccer teamwork challenge as being about where the player with the ball should pass or dribble, but at least as important is where the agents position themselves when they *do not* have the ball [10]. Positioning the players in a formation requires the agents to coordinate with each other and determine where each agent should position itself on the field. In our team, players' roles are determined in three steps. First, a full team formation is computed; second, each player computes the best assignment of players to roles in this formation according to its own view of the world; and third, a coordination mechanism is used to communicate and choose among all players' suggestions. In this section, we use the terms (player) position and (player) role interchangeably.

6.1 Formation

In general, the team formation is determined by the ball position on the field. As an example, Figure 5 depicts the different role positions of the formation and their relative offsets when the ball is at the center of the field. As can be seen in the figure, the formation can be broken up into two separate groups, an offensive and a defensive group. Within the offensive group, the role positions on the field are determined by adding a specific offset to the ball's coordinates. The *onBall* role, assigned to the player closest to the ball,

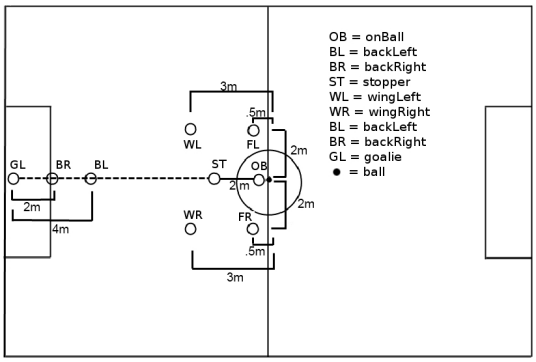


Figure 5: Formation role positions.

is always based on where the ball is and is therefore never given an offset. On either side of the ball we have two forward roles, *forwardRight* and *forwardLeft*. Directly behind the ball we have a *stopper* role as well as two additional roles, *wingLeft* and *wingRight*, located behind and to either side of the ball. When the ball is near the edge of the field we adjust some of the roles' offsets from the ball so as to prevent them from moving outside the field of play.

Within the defensive group there are two roles, *backLeft* and *backRight*. To determine their position on the field a line is calculated between the center of our goal and the ball. Both backs are placed along that line at specific offsets from the end line. The goalie positions itself independently of its teammates in order to always be in the best position to dive and stop a shot on goal. If the goalie assumes the *onBall* role, however, a third role is included within the defensive group, the *goalie* role. A field player assigned to the *goalie* role is told to stand in front of the center of the goal to cover for the goalie going to the ball.

6.2 Assigning Agents to Roles

Given a desired team formation, we need to map players to roles (target positions on the field). A naïve mapping having each player permanently mapped to one of the roles performs poorly due to the dynamic nature of the game. With such static roles an agent assigned to a defensive role may end up out of position and, without being able to switch roles with a teammate in a better position to defend, allow for the opponent to have a clear path to the goal. In this section, we present a dynamic role assignment algorithm. A role assignment algorithm can be thought of as implementing a role assignment *function*, which takes as input the state of the world, and outputs a one-to-one mapping of players to roles. We start by defining three properties that a role assignment function must satisfy (Section 6.2.1). We then construct a role assignment function that satisfies these properties (Section 6.2.2). Finally, we present a dynamic programming algorithm implementing this function (Section 6.2.3).

6.2.1 Desired Properties of a Valid Role Assignment Function

Before listing desired properties of a role assignment function we make a couple of assumptions. The first of these is that no two agents and no two role positions occupy the same position on the field. Secondly we assume that all agents move toward fixed role positions along a straight line

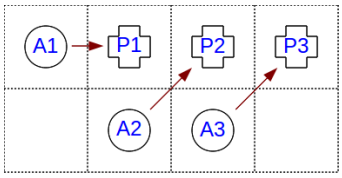


Figure 6: Lowest lexicographical cost (shown with arrows) to highest cost ordering of mappings from agents (A1,A2,A3) to role positions (P1,P2,P3). Each row represents the cost of a single mapping.

1:	$\sqrt{2}$ (A2→P2),	$\sqrt{2}$ (A3→P3),	1 (A1→P1)
2:	2 (A1→P2),	$\sqrt{2}$ (A3→P3),	1 (A2→P1)
3:	$\sqrt{5}$ (A2→P3),	1 (A1→P1),	1 (A3→P2)
4:	$\sqrt{5}$ (A2→P3),	2 (A1→P2),	$\sqrt{2}$ (A3→P1)
5:	3 (A1→P3),	1 (A2→P1),	1 (A3→P2)
6:	3 (A1→P3),	$\sqrt{2}$ (A2→P2),	$\sqrt{2}$ (A3→P1)

at the same constant speed. While this assumption is not always completely accurate, the omnidirectional walk described in Section 4 gives a fair approximation of constant speed movement along a straight line.

We call a role assignment function *valid* if it satisfies the following three properties:

1. *Minimizing longest distance* - it minimizes the maximum distance from a player to target, with respect to all possible mappings.
2. *Avoiding collisions* - agents do not collide with each other as they move to their assigned positions.
3. *Dynamically consistent* - a role assignment function f is dynamically consistent if, given a *fixed* set of target positions, if f outputs a mapping m of players to targets at time T, and the players are moving towards these targets, f would output m for every time $t > T$.

Based on our two given assumptions, the first two properties guarantee that the chosen role assignment is one that minimizes the time to its completion, and the third property guarantees that once a role assignment is decided, it is unchanged as long as the target positions are not changed.

6.2.2 Constructing a Valid Role Assignment Function

Let M be the set of all one-to-one mappings between players and roles. If the number of players is n, then there are $n!$ possible such mappings. Given a state of the world, specifically n player positions and n target positions, let the *cost* of a mapping m be the n-tuple of distances from each player to its target, sorted in decreasing order. We can then sort all the $n!$ possible mappings based on their costs, where comparing two costs is done lexicographically. Sorted costs of mappings from agents to role positions for a small example are shown in Figure 6.

Denote the role assignment function that always outputs the mapping with the lexicographically smallest cost as f_v. Here we provide an informal proof sketch that f_v is a valid role assignment; we provide a longer, more thorough derivation in a technical report [11].

THEOREM 1. *f_v is a valid role assignment function.*

It is trivial to see that f_v minimizes the longest distance traveled by any agent (Property 1) as the lexicographical ordering of distance tuples sorted in descending order ensures

this. If two agents in a mapping are to collide (Property 2) it can be shown, through the triangle inequality, that f_v will find a lower cost mapping as switching the two agents' targets reduces the maximum distance either must travel. Finally, as we assume all agents move toward their targets at the same constant rate, the distance between any agent and target will not decrease any faster than the distance between an agent and the target it is assigned to. This serves to preserve the lowest cost lexicographical ordering of the chosen mapping by f_v across all timesteps thereby providing dynamic consistency (Property 3). The next section presents an algorithm that implements f_v.

6.2.3 Dynamic Programming Algorithm for Role Assignment

Clearly f_v could be calculated using a brute force method to compare all possible mappings. As there are 8 field players, this would require creating $8! = 40,320$ mappings, then computing the cost of each of the mappings, and finally sorting them lexicographically and choosing the smallest one. However, as our agent acts in real time, and f_v needs to be computed during a decision cycle (0.02 seconds), a brute force method is too computationally expensive. Therefore, we present a dynamic programming implementation shown in Algorithm 1 that is able to compute f_v within the time constraints imposed by the decision cycle's length.

Algorithm 1 Dynamic programming implementation

```
1: HashMap bestRoleMap = ∅
2: Agents = {a_1, ..., a_n}
3: Positions = {p_1, ..., p_n}
4: for k = 1 to n do
5:    for each a in Agents do
6:       S = (n-1 choose k-1) sets of k-1 agents from Agents - {a}
7:       for each s in S do
8:          Mapping m_0 = bestRoleMap[s]
9:          Mapping m = (a -> p_k) ∪ m_o
10:         bestRoleMap[a ∪ s] = mincost(m, bestRoleMap[a ∪ s])
11: return bestRoleMap[Agents]
```

THEOREM 2. *Let A and P be sets of n agents and positions respectively. Denote the mapping $m := f_v(A, P)$. Let m_0 be a subset of m that maps a subset of agents $A_0 \subset A$ to a subset of positions $P_0 \subset P$. Then m_0 is also the mapping returned by $f_v(A_0, P_0)$.*

A key recursive property of f_v that allows us to exploit dynamic programming is expressed in Theorem 2. This property stems from the fact that if within any subset of a mapping a lower cost mapping is found, then the cost of the complete mapping can be reduced by augmenting the complete mapping with that of the subset's lower cost mapping. The savings from using dynamic programming comes from only evaluating mappings whose subset mappings are returned by f_v. This is accomplished in Algorithm 1 by iteratively building up optimal mappings for position sets from $\{p_1\}$ to $\{p_1, ..., p_n\}$, and using optimal mappings of $k-1$ agents to positions $\{p_1, ..., p_{k-1}\}$ (line 8) as a base when constructing each new mapping of k agents to positions $\{p_1, ..., p_k\}$ (line 9), before saving the lowest cost mapping for the current set of k agents to positions $\{p_1, ..., p_k\}$ (line 10).

As $\binom{n-1}{k-1}$ agent subset mapping combinations are evaluated for mappings of each agent assigned to the kth position, the total number of mappings computed for each of the n agents is thus equivalent to the sum of the $n-1$ binomial coefficients. That is,

$$\sum_{k=1}^{n} \binom{n-1}{k-1} = \sum_{k=0}^{n-1} \binom{n-1}{k} = 2^{n-1}$$

Therefore the total number of mappings that must be evaluated using our dynamic programming approach is $n2^{n-1}$. For $n = 8$ we thus only have to evaluate 1024 mappings which is very manageable.

Table 1: Full game results, averaged over 100 games. Each row corresponds to an agent with varying formation and positioning systems as described in Section 6.3. Entries show the goal difference from 10 minute games versus our agent using the dynamic role positioning system and formation described in Section 6. Values in parentheses are the standard error.

Team	Goal Difference
Defense	.29 (.06)
Static	.32 (.07)
AllBall	.43 (.09)
Boxes	1.26 (.10)

6.3 Formation Evaluation

To test how our formation and role positioning system affects the team's performance we created a number of teams to play against by modifying the positioning system of UT Austin Villa that was used in the competition.

AllBall No formations and every agent except for the goalie just goes to the ball.

Static Each role is statically assigned to an agent based on its uniform number.

Defense Defensive formation in which only two agents are in the offensive group (one on the ball and the other directly behind the ball)

Boxes Field is divided into fixed boxes and each agent is dynamically assigned to a home position in one of the boxes. Similar to the positioning system used in [14].

Results of UT Austin Villa playing against these modified versions of itself are shown in Table 1. We see that a very defensive formation used by the *Defense* agent hurts performance a little likely because the best defense is a good offense. Dynamically assigning roles is better than statically fixing them as is clear in the degradation in performance of the *Static* agent. Having and maintaining formations is also important which is evident by the positive goal difference recorded when playing against the *AllBall* agent. The poor performance of the *Boxes* agent, in which the positions on the field are somewhat static and not calculated as relative offsets to the ball, underscores the importance of being around the ball and adjusting positions on the field based on the current state of the game.

7. COMPETITION RESULTS

UT Austin Villa 2011 won all 24 of its games during the RoboCup 2011 3D simulation competition, scoring 136 goals and conceding none. Even so, competitions of this sort do not consist of enough games to validate that any team is better than another by a statistically significant margin. In

Table 2: Full game results, averaged over 100 games. Each row corresponds to an agent from the RoboCup 2011 competition, with its rank therein achieved. Entries show the goal difference from 10 minute games versus our final optimized agent. Values in parentheses are the standard error.

Rank	Team	Goal Difference
3	apollo3d	1.45 (.11)
5-8	boldhearts	2.00 (0.11)
5-8	robocanes	2.40 (0.10)
2	cit3d	3.33 (0.12)
5-8	fcportugal3d	3.75 (0.11)
9-12	magmaoffenburg	4.77 (0.12)
9-12	oxblue	4.83 (0.10)
4	kylinsky	5.52 (0.14)
9-12	dreamwing3d	6.22 (0.13)
5-8	seuredsun	6.79 (0.13)
13-18	karachikoalas	6.79 (0.09)
9-12	beestanbul	7.12 (0.11)
13-18	nexus3d	7.35 (0.13)
13-18	hfutengine3d	7.37 (0.13)
13-18	futk3d	7.90 (0.10)
13-18	naoteamhumboldt	8.13 (0.12)
19-22	nomofc	10.14 (0.09)
13-18	kaveh/rail	10.25 (0.10)
19-22	bahia3d	11.01 (0.11)
19-22	l3msim	11.16 (0.11)
19-22	farzanegan	11.23 (0.12)

order to validate the results of the competition, in Table 2 we show the performance of our team when playing 100 games against each of the other 21 teams' released binaries from the competition. UT Austin Villa won by at least an average goal difference of 1.45 against every team. Furthermore, of these 2100 games played to generate the data for Table 2, our agent won all but 21 of them which ended in ties (no losses). The few ties were all against three of the better teams: apollo3d, boldhearts, and robocanes. We can therefore conclude that UT Austin Villa was the rightful champion of the competition.

While there were multiple factors and components that contributed to the success of UT Austin Villa in winning the competition, its omnidirectional walk was the one which proved to be the most crucial. When switching out the omnidirectional walk developed for the 2011 competition with the fixed directional walk used in the 2010 competition, and described in [15], the team did not fare nearly as well. The agent with the previous year's walk had a negative average goal differential against nine of the teams from the 2011 competition, suggesting a probable tenth place finish. Also this agent lost to our 2011 agent by an average of 6.32 goals across 100 games with a standard error of .13

8. SUMMARY AND DISCUSSION

We have presented the architecture and key components of the UT Austin Villa 2011 RoboCup 3D simulation league team. These key components include an omnidirectional walk engine and associated walk parameter optimization framework, an inverse kinematics based kicking architecture, and a dynamic role and formation positioning system.

Our ongoing research agenda includes applying what we have learned in simulation to the actual Nao robots which we use to compete in the Standard Platform league of RoboCup. For next year's competition we expect to better integrate and utilize our kicking system in order to improve the performance of the team. Additionally, we would like to learn and add further parameter sets to our team's walk engine for important subtasks such as goalie positioning to get ready to block a shot.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. Thanks especially to UT Austin Villa 2011 team members Michael Quinlan, Nick Collins, and Art Richards. Also thanks to Yinon Bentor and Suyog Dutt Jain for contributions to early versions of the optimization framework employed by the team. LARG research is supported in part by NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030). Patrick MacAlpine and Samuel Barrett are supported by NDSEG fellowships.

9. REFERENCES

[1] Aldebaran Humanoid Robot Nao. `http://www.aldebaran-robotics.com/eng/`.
[2] Open Dynamics Engine. `http://www.ode.org/`.
[3] RoboCup. `http://www.robocup.org/`.
[4] SimSpark. `http://simspark.sourceforge.net/`.
[5] E. Angel. *Interactive Computer Graphics*. Pearson Education, Inc., 5th edition, 2009.
[6] S. Behnke, M. Schreiber, J. Stückler, R. Renner, and H. Strasdat. See, walk, and kick: Humanoid robots start to play soccer. In *Proc. of the 6th IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids 2006)*, pages 497–503. IEEE, 2006.
[7] R. Diankov and J. Kuffner. Openrave: A planning architecture for autonomous robotics. Technical Report CMU-RI-TR-08-34, Robotics Institute, Pittsburgh, PA, July 2008.
[8] C. Graf, A. Härtl, T. Röfer, and T. Laue. A robust closed-loop gait for the standard platform league humanoid. In *Proc. of the 4th Workshop on Humanoid Soccer Robots in conjunction with the 2009 IEEE-RAS Int. Conf. on Humanoid Robots*, pages 30 – 37, 2009.
[9] N. Hansen. *The CMA Evolution Strategy: A Tutorial*, January 2009. `http://www.lri.fr/~hansen/cmatutorial.pdf`.
[10] S. Kalyanakrishnan and P. Stone. Learning complementary multiagent behaviors: A case study. In *RoboCup 2009: Robot Soccer World Cup XIII*, pages 153–165. Springer, 2010.
[11] P. MacAlpine, D. Urieli, S. Barrett, S. Kalyanakrishnan, F. Barrera, A. Lopez-Mobilia, N. Ştiurcă, V. Vu, and P. Stone. UT Austin Villa 2011 3D Simulation Team report. Technical Report AI11-10, The Univ. of Texas at Austin, Dept. of Computer Science, AI Laboratory, December 2011.
[12] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–73, 2009.
[13] P. Stone. *Layered Learning in Multi-Agent Systems*. PhD thesis, School of Computer Science, Carnegie Mellon Univ., Pittsburgh, PA, USA, December 1998.
[14] P. Stone and M. Veloso. Task decomposition, dynamic role assignment, and low-bandwidth communication for real-time strategic teamwork. *Artificial Intelligence*, 110(2):241–273, June 1999.
[15] D. Urieli, P. MacAlpine, S. Kalyanakrishnan, Y. Bentor, and P. Stone. On optimizing interdependent skills: A case study in simulated 3D humanoid robot soccer. In *Proc. of the Tenth Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2011)*, pages 769–776, May 2011.

Session 4A
Robotics II

Property-driven design for swarm robotics

Manuele Brambilla, Carlo Pinciroli, Mauro Birattari and Marco Dorigo
Université Libre de Bruxelles
Brussels, Belgium
{mbrambil,cpinciro,mbiro,mdorigo}@ulb.ac.be

ABSTRACT

In this paper, we propose a novel top-down design method for the development of collective behaviors of swarm robotics systems called *property-driven design*. Swarm robotics systems are usually designed and developed using a *code-and-fix* approach, that is, the developer devises, tests and modifies the individual robot behaviors until a desired collective behavior is obtained. The code-and-fix approach can be very time consuming and relies completely on the ingenuity and expertise of the designer. The idea of property-driven design is that a swarm robotics system can be described by specifying formally a set of desired properties. In an iterative process similar to test-driven development, the developer produces a model of the system that satisfies the desired properties. Subsequently, the system is implemented in simulation and using real robots. Property-driven design helps to minimize the risk of developing a system that does not satisfy the required properties, and to promote the reuse of hardware independent models. In this paper, we start by giving a general description of the method. We then present a possible way to apply it by using Discrete Time Markov Chains (DTMC) and Probabilistic Computation Tree Logic* (PCTL*). Finally, we conclude by presenting the application of the proposed method to the design and development of a swarm robotics system performing aggregation.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Robotics

General Terms

Design, Verification

Keywords

Swarm robotics, Swarm engineering, Top-down design, Aggregation

1. INTRODUCTION

Swarm robotics is a distributed approach to multi-robot systems in which, through local interactions, robots achieve a self-organized collective behavior. Swarm robotics systems have the potential to display interesting properties, such as robustness, scalability and flexibility [13].

Swarm robotics systems are complex systems [9]. They exhibit dynamics at two different levels: the collective, or macroscopic, level, and the individual, or microscopic, level. The collective behavior is the result of the interactions of the individual robots with each other and with the environment. In order to obtain a desired collective behavior, the individual behaviors and the interactions of the robots must be carefully designed. However, this design process is usually non-trivial, as the dynamics of complex systems are very often difficult to predict [1].

Despite the increasing attention on swarm robotics systems in the past two decades [4], a top-down methodology for the design and development of this kind of systems has not been defined yet. Swarm robotics systems are usually designed and developed using a *code-and-fix* approach [7]. This means that, usually, the individual behavior is developed, tested and modified until the desired collective behavior is obtained. This process is often performed first in computer simulations and eventually on real robots (see Fig. 1).

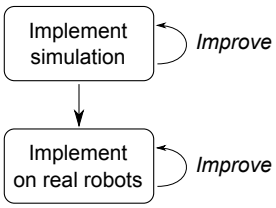


Figure 1: The most common method for the development of a swarm robotics system. The individual behavior is developed, tested and modified until the desired collective behavior of the swarm is obtained.

We believe that, to improve the quality of swarm robotics systems and to reduce the effort for their development, it is necessary to create a new, specific branch of engineering that we call *swarm engineering*. We define swarm engineering to be the systematic application of scientific and technical knowledge to specify requirements, design, realize, verify, validate, operate and maintain an artificial swarm intelligence system.

Traditional system engineering approaches are not suited for swarm robotics systems. System engineering is mainly aimed towards centralized systems or, in general, systems in which the interactions of the components can be precisely

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

predicted. In this respect, swarm robotics systems, which can have hundreds of interacting robots, present unprecedented challenges [12].

In this paper, we propose a top-down design method, that we call *property-driven design*. Property-driven design provides ways to specify requirements, design, develop, verify and validate a swarm robotics system.

We believe that property-driven design has many advantages compared with code-and-fix development: it helps to formally specify the requirements of the system; to reduce the risk of developing the "wrong" system, that is, a system that does not satisfy the requirements; to develop a set of hardware independent models that can be reused for future applications; and to shift the focus of the development process from implementation to design, given that most of the developing effort happens in the modeling phase.

In Section 2, we present related work on top-down design methods and verification techniques for swarm robotics. In Section 3, we present property-driven design. In Section 4, we propose a possible way to specify properties and validate a model for a swarm robotics system. In Section 5, we present an example application of property-driven design to the design and development of a system able to perform aggregation. In Section 6, we discuss about the features and limits of property-driven design. In Section 7, we conclude this paper.

2. RELATED WORK

Design methods: Developing a top-down design method for complex systems is still an open challenge [31]. In the last years, the effort on this topic has been quite limited.

Bachrach et al. [2] proposed a scripting language called *Protoswarm*. This language enables the definition of a vector field on an abstract spatial machine. This vector field is then translated by a middleware into individual robot behaviors. Protoswarm allows the developer to focus mostly on the collective behavior, removing some of the effort necessary to develop the individual behaviors. However, Protoswarm is thought for situations in which the robots are covering the entire environment and keep constant network connectivity. Thus, it is more suited for sensor networks than for swarm robotics systems. Another thing to note is that Protoswarm is not a design method, but a scripting language. As such, it can be used only as a development tool, requiring an appropriate design method to guide the process.

Kazadi et al. [19] proposed *the Hamiltonian method*, a design method based on Hamiltonian vector fields. This method allows one to develop systems by specifying one or more numerical properties, such as the energy level of a particular state of the system. One limitation of this design method is that it is suited only for spatially-organizing behaviors. The goal of spatially-organizing behaviors is to achieve a specific robot distribution in the environment, such as pattern formation (e.g., [30]).

Another possible approach to the design of swarm robotics systems are automatic design methods. Work on automatic design methods for swarm robotics systems focuses mainly on evolutionary robotics [24] and reinforcement learning [26]. Automatic design methods can be considered top-down approaches because, in principle, the development process is driven by the desired collective-level goal behavior. However, a lot of domain knowledge is required to tackle medium to complex applications. Moreover, once a system is obtained, it is, in general, non-trivial to understand its behavior and it is often very difficult to verify its properties or adapt it to other applications, even if they are similar to the original one.

Property verification: The problem of property verification in swarm robotics systems has been tackled only in a limited way. Dixon et al. [14], used Linear Temporal Logic (LTL) to define properties of individual robots and of the swarm. This method is based on modeling the individual robot behavior with a Markov chain, and then considering the collective behavior as the result of the *and*-composition of these individual-level models. A limitation of this approach is that linear temporal logic deals only with binary values. This limits the possibility to analyze systems displaying stochastic properties, such as non-trivial swarm robotics systems. Furthermore, in this method, there is a possible scalability problem, because the number of states of the system grows exponentially with the number of robots: $\sim \Theta(k^n)$, where k is the number of states of the individual Markov chain and n is the number of robots.

Recently, Konur et al. [20] proposed an approach to verify formally the properties of a swarm behavior through *probabilistic computation tree logic* [16]. Their approach is able to overcome the limits of linear temporal logical while providing scalability.

3. PROPERTY-DRIVEN DESIGN

The idea behind property-driven design is that a swarm robotics system can be formally described through a series of properties. These properties are the distinguishing features of the system the developer wants to realize. They can be task specific, such as *the system eventually completes the task X*, or they can express more generic properties, such as *the system keeps working as long as there are at least N robots* or *the system will never enter state Y*.

A schema showing the different steps of property-driven design is presented in Figure 2.

Phase One: The first phase of property-driven design consists in formally specifying the requirements of the system by stating its desired properties. The clearer and more complete these properties are in this phase, the more the developed system will conform to the requirements.

Phase Two: In the second phase, a model of the system is created. At first, similarly to test-driven development [5], one cannot expect the system to satisfy all the desired properties. In an iterative process, the developer expands and improves the model, and checks whether the properties are verified. The outcome of this process is a model of the system that satisfies the stated properties. Note that the model must be complete just enough to capture all the important characteristics of the system, avoiding unnecessary complication. For example, to model failures, the developer could insert in the model only a general *failure* state, without specifying all the possible hardware problems if not necessary. Eventually, through this process, one obtains a model that satisfies all the required properties.

At the end of this phase, the developed model is robot independent. The developer can now identify a set of necessary sensors and actuators to select the proper robot platform to use in implementing the system.

Phase Three: In this phase, the developer can use the model to guide the process of implementing the swarm

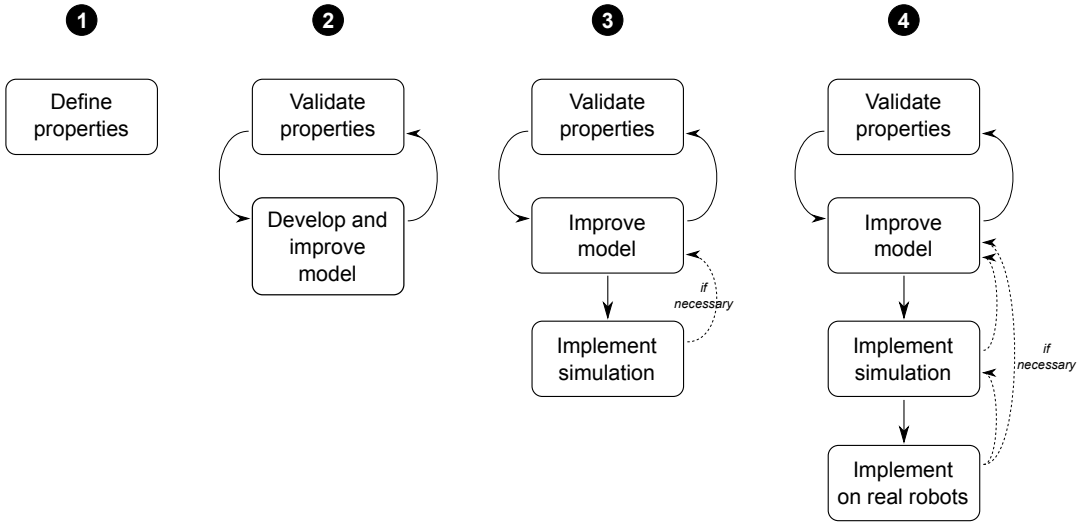


Figure 2: The four different phases of property-driven design to develop a swarm robotics system.

robotics system using, for instance, a physics-based computer simulation (henceforth simply simulation). As discussed in Section 1, this phase can be challenging, as moving from a macroscopic model to the microscopic implementation is a process that is guided mainly by the ingenuity and expertise of the developer. However, the defined model gives a clear picture of the system that can greatly help in its development.

It is possible that the simulation does not validate the model [22]. In this case the developer must go back to step 2, modify the model to include the results obtained from the simulation, and verify whether the required properties still hold true.

Phase Four: The last phase consists in deploying the system on real robots.

Similarly to the transition between the model and the simulation, if the implementation on real robots reveals that some assumptions made during the previous phases do not hold, it might be necessary to modify the simulated version or the model, in order to keep all levels consistent.

4. DTMC AND PCTL*

So far, we purposely did not mention how to model the system or how to specify its properties. There are several possible ways to perform this activity. Here, we do not discuss the different options available, as a review of the possible techniques for modeling swarm robotics system is out of the scope of this paper. The interested reader can refer to Lerman et al. [21].

Of all the various possibilities, the developer can choose the one that best fits the system to develop and its personal experience. In this section, we briefly introduce one possible way to model a swarm robotics system and specify its properties based on Deterministic Time Markov Chains (DTMC) and Probabilistic Computation Tree Logic* (PCTL*).

DTMC are often used to model swarm robotics systems [21]. One of the main advantages of DTMC is that, in many cases, the model comprises both the microscopic and the macroscopic levels. At the microscopic level, the model represents the behavior of a single robot. At the macroscopic level, each state can be used to count the number of robots in that particular state. For example, at the microscopic level, one can model the behavior of a single robot with a 3-state DTMC. The same DTMC can be augmented by associating a counter to each state. Each counter keeps track of the number of robots in the associated state. Another advantage of DTMC is that their use can ease property verification, especially if the properties are written through the use of logic predicates [11].

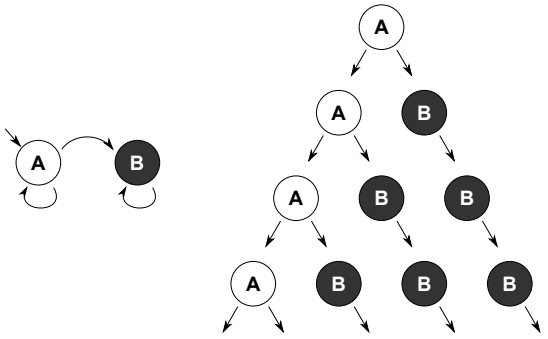


Figure 3: A simple Markov chain (on the left) and its computation tree (on the right).

Among the many formal logical systems, we consider Probabilistic Computation Tree Logic* (PCTL*). PCTL*, originally developed by Hansson and Jonsson [16], is an extension of CTL (Computation Tree Logic), a branching time logic. CTL is based on the idea that a Markov chain can be "expanded" in a computation tree. A computation tree is a potentially infinite rooted tree in which the root is the initial state of the corresponding Markov chain, and each node is a possible state of the system. The edges link a state with its next possible states. An example of a simple Markov chain and its computation tree is displayed in Figure 3.

Through CTL, one can express time-related properties

such as *property α will eventually become true* or *property α will hold true for at least 10 seconds*. PCTL* extends CTL by introducing probabilities. In this way, one can express properties such as *property α will eventually become true with probability 0.45* or *there is a 0.7 probability that α will hold true for 10 seconds*. PCTL* is well suited for swarm robotics systems as it can capture the time-related and stochastic aspects of this kind of systems. We do not discuss the details of the presented logics, we refer the interested reader to Ciesinski and Größer [10].

Our approach is based on model checking, a technique that allows to verify automatically and completely whether a set of formulae is satisfied by a given system. As model checker software we choose PRISM [17]. PRISM is a probabilistic model checker which supports DTMC and PCTL* among many other models and logics. With PRISM, it is possible not only to verify properties, but also to perform so called "experiments," in which the model checker computes the probability of the property being true against different parameter values. In this way, it is possible to find the parameter set that scores the best probability in verifying a property.

5. AN EXAMPLE APPLICATION: AGGREGATION

In order to show the characteristics of property-driven design, we present an example application: *aggregation*. In this application the robots have to cluster in an area of the environment. The robots have neither a map of the environment nor knowledge of the position of the other robots. We choose aggregation as a case study for four reasons:

- aggregation is a simple behavior: this allows us to focus on the development process without being hampered by the details of the system itself;
- aggregation is a common test-case behavior for the swarm robotics community, and it has been studied extensively in the past (see, for example, [3, 22, 29, 32, 6]);
- aggregation is a collective behavior that cannot be developed easily with Protoswarm [2], since the robots can often lose network connectivity; or using the Hamiltonian method [19], since no specific spatial distribution is required once the aggregate is formed;
- aggregation possesses many of the salient traits of a typical swarm robotics behavior. It is completely distributed, it is based on simple robot-to-robot interactions, and it is characterized by stochasticity and spatial requirements.

The collective behavior we study in this paper is similar to the one presented by Jeanson et al. [18]. We consider a dodecagonal environment with two black spots called *area A* and *area B*. We call *area C* the remaining white area. Each of the black spots is big enough to host all the robots. See Figure 7 for a screenshot of the environment. In the following, we will follow the 4-phase process explained in Section 3 using DTMC and PTCL* as modeling tools and PRISM as model checker.

Phase One: The property we focus on is *"eventually all the robots form an aggregate"*. We would like that the aggregate is formed as fast as possible (for example, in the first 1,000 seconds) but we do not differentiate between obtaining the aggregate in area A or area B. Using PRISM syntax, we can define this property as follows:

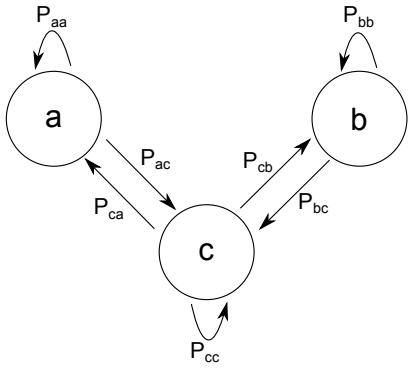


Figure 4: The DTMC model of the aggregation example. Each state is used to count the number of robots in that particular area.

```
P=? [F<=1000 (a=N_total)|(b=N_total)]        (1)
```

In less formal terms, we compute the probability (`P=?`) that, in the first thousand seconds (`F<=1000`), the number of robots in area A or in area B is equal to the total number of robots in the swarm (`(a=N_total)|(b=N_total)`). Since we want to maximize this probability, we do not specify a value for it.

Another property we want for the system is that the aggregate, once formed, is stable for a certain period. In this example we set such period to 10 seconds. We verify this property with probability greater or equal to $\frac{2}{3} \simeq 0.67$:

```
(a=N_total)|(b=N_total) =>
P>=0.67 [G>=10 (a=N_total)|(b=N_total)]        (2)
```

In natural language, Property 2 can be expressed in this way: from the aggregate state (`(a=N_total)|(b=N_total)`) is it true with probability greater or equal to 0.67 (`=> P>=0.67`) that the system stays for at least 10 seconds (`G>=10`) in the aggregate state?

Phase Two: Once the above desired properties have been specified we need to build the model. We start by setting the total number of robots in the system. In order to perform a scalability test, we selected three different group sizes: $N_t = 10, 20, 50$[1]. Then, we specify three states: state S_a, S_b and S_c. A robot in area A or B is in state S_a or S_b, respectively. Robots outside area A or B are in state S_c. Moreover, three counters `a`, `b` and `c` are associated to the respective states. These counters are used to keep track of the number of robots that are in state S_a, S_b and S_c, respectively. Note that `a+b+c`=N_t. See Figure 4 for the DTMC model of the system.

We design the following behavior for a robot: it performs random walk and when it finds a black area it stops with probability 1. The robot then decides whether to leave according to a certain probability.

[1]We tested the system also with 100 robots, but the probability of obtaining an aggregate in less than 1000 seconds was close to 0. We decided thus not to include these results in order to simplify the explanation.

In this initial stage of the definition of the model, we assume that our system can be effectively described by a *non-spatial* model, that is, a model in which the trajectories of the robots are ignored and a robot can move instantaneously from area C to area A or B, and vice versa. Moreover, we also ignore the effects of interferences between robots [21]. However, especially for larger group sizes, the performance of the system may be reduced by the fact that robots must avoid each other or that robots stopping in the black areas prevent other robots from entering it. In case these assumption proves to be not realistic and the results obtained with the model do not match those obtained in simulation or with real robots, we will modify them in the following phases, as explained in Section 3. Note that a model of a similar system is presented in O'Grady et al. [25].

Since our model is non-spatial and ignores interference, we consider only the geometric properties of the areas to compute p_{ca}. A robot in area C can either go in area A, go in area B or stay in area C. This means that a robot in area C has a probability of going from area C to area A equal to $p_{ca} = \frac{A_A}{A_{arena}}$, of going from area C to area B equal to $p_{cb} = \frac{A_B}{A_{arena}}$, and of staying in area C equal to $p_{cc} = \frac{A_C}{A_{arena}} = 1-(p_{ca}+p_{cb})$. In our scenarios we used three different arena sizes for the three different group sizes. In Table 1 it is possible to find the details about the parameters used for the experiments.

We need to define the remaining probabilities. The aggregate can be obtained in area A or area B, thus we set the probabilities of leaving these two areas to be equal: $p_{ac} = p_{bc}$. Since the two areas have the same size we set $p_{aa} = p_{bb}$. A robot in area A can only return to area C or stay in area A, thus $p_{aa} = 1 - p_{ac}$. The only independent probability remaining is p_{ac}. Initially, we set p_{ac} to a fixed value. Through model checking, we can find the value of p_{ac} that maximizes the probability of satisfying Property 1. The process consists in automatically testing the model for different p_{ac} values and find the best one.

The best values found with PRISM are $p_{ac} = 0.05, 0.04$, 0.04 when $N_t = 10, 20, 50$, respectively. With these values, the probabilities of satisfying Property 1 are 0.75, 0.15 and 8.8×10^{-5}. Property 2 is not satisfied for any of the three group sizes. The developed behavior, thus, obtains poor results and the system does not cope well with increasing group sizes.

It is thus necessary to improve the developed model by modifying the behavior of the robots. A fixed p_{ac} does not promote the formation of a single cluster. A better solution is to let a robot decide whether to leave according to the number of sensed robots around it [18]: with only few robots nearby, the probability to leave the aggregate p_{ac} is high and vice versa. We set $p_{ac} = p_{min-ac} * (N_s + 1)$, where p_{min-ac} is the minimum staying probability we want for a robot and N_s is the number of other robots sensed. We add 1 to the number of robots sensed, as we want to include also the robot that is choosing its next action. Subsequently, we use PRISM to find the best value of p_{min-ac} for the different group sizes. As reported in Table 1, results are better than before, both for Property 1 and Property 2.

With the current model we are also able to define requirements on the hardware capabilities of the robots: a ground sensor, to differentiate between the two black areas A and B and the white area C; a sensor to detect nearby robots; and wheels to move. An example of such a robot is the e-puck [23] robot which has a range and bearing board that allows it to perceive the presence of neighboring robots [15].

Table 1: A table that presents the obtained results. Column p_{min-ac} shows the best value of p_{min-ac} found using PRISM. Column Pr 1 shows the probability of satisfying Property 1, and column Pr 2 shows whether Property 2 is satisfied.

N_t	A_A	A_{arena}	p_{ca}	p_{min-ac}	Pr 1	Pr 2
10	$0.38m^2$	$4.91m^2$	0.0784	[0.19, 0.24]	0.95	✓
20	$0.78m^2$	$19.63m^2$	0.0625	0.12	0.79	✓
50	$3.14m^2$	$50.26m^2$	0.0625	0.10	0.25	✓

Phase Three: In this aggregation example, the model captures well the microscopic behavior of the single robots, thus it is quite easy to implement the system in simulation. However, several implementation details are not explicitly present in the model, such as how the robots perform random walk. These implementation details must be dealt with in such a way that they do not falsify the model.

We implemented the system using the ARGoS simulator [27]. Figure 6 shows a screenshot of the simulated system. We performed three different sets of experiments, one for each group size. To validate the model we measured the average time necessary to form a complete aggregate on 100 runs with different values of p_{min-ac}. The robots were deployed in a random position at the beginning of each experiment. Each experiment stopped when a complete aggregate was formed or after 10,000 seconds.

As reported in Figure 5, for all the three group sizes, the best results were obtained with the value p_{min-ac} predicted using the model. However, the results obtained for Property 1 with the simulated version of the system are usually worse than those predicted by the model, in particular with 20 and 50 robots. With 10 robots and $p_{min-ac} = 0.22$ the simulated system was able to form a complete aggregate before 1,000 seconds 100 times out of 100, in line with the model predictions. However, with 20 robots and $p_{min-ac} = 0.12$, Property 1 was satisfied only 53 times out of 100, whereas in the model it was satisfied with a probability of 0.79. With 50 robots and $p_{min-ac} = 0.10$ the difference is even more evident: only 2 runs out of 100 resulted in an aggregation time of under 1,000 seconds whereas the model predicted a probability of satisfying Property 1 of 0.25.

As explained in Section 3, since the results obtained from the model and from the simulation do not match, we need to modify the model in order to make them consistent. The discrepancy between the model and the simulated system is due to the fact that, as the number of robots grows, interference between robots reduces p_{ca}. This is because the robots spend more time avoiding collisions and because the robots stopping in the black areas prevent other robots from accessing them. Reducing p_{ca} in the model allows us to obtain results that are closer to those obtained in simulation. For 20 robots and $p_{ca} = 0.0475$, we observe that Property 1 is satisfied with probability 0.5275, which matches the results obtained in simulation. For 50 robots we set $p_{ca} = 0.041$, which gives a probability of satisfying Property 1 of 0.01.

We also tested Property 2. 100 runs of the simulated experiments were executed for 10,000 seconds with the three

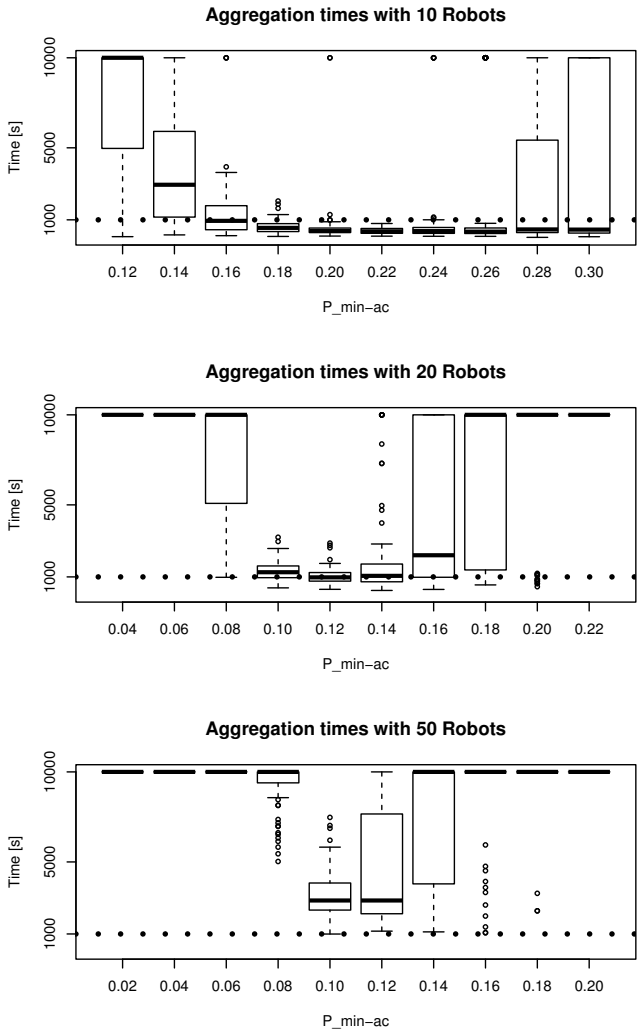


Figure 5: Some results obtained with the ARGoS simulator. The graphs show the time necessary to form the complete aggregate with different p_{min-ac} over 100 runs for 10, 20 and 50 robots.

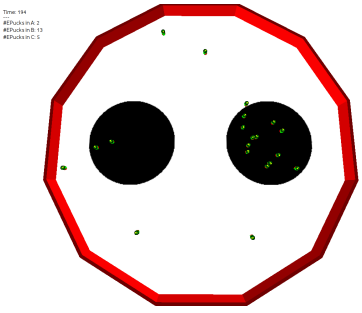


Figure 6: A screenshot of the simulated version of the system using 20 robots.

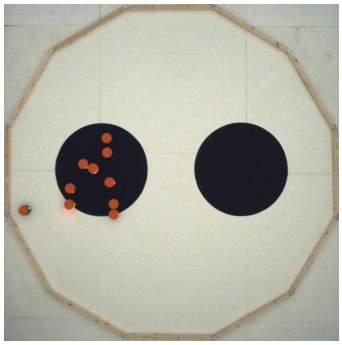


Figure 7: A screenshot of the experiment performed with the 10 e-puck robots.

group sizes. In the experiments, we measured whether the system satisfies Property 2, that is, whether a complete aggregate, once formed, lasts more than 10 seconds. In all the cases in which a complete aggregate was formed before 10,000 seconds, Property 2 was satisfied.

Videos of the simulated experiments are available in the supplementary pages [8].

Phase Four: In the last phase, we implement the system using real e-pucks. We performed 10 experiments with a group of 10 e-pucks in an arena identical to the simulated one. A picture of an experiment can be seen in Figure 7. Figure 8 shows a comparison between the results obtained with the real robots and in simulation. A video of a run is available in the supplementary pages [8].

In 10 runs out of 10, both Property 1 and Property 2 were satisfied. The results obtained with the real robots are in line with those obtained in simulation, even though the aggregation time is slightly longer. This is probably due to a higher wheel speed in the simulated experiments.

6. DISCUSSION

Property-driven design aims at supporting the development of swarm engineering, that is, a systematic application of scientific and technical knowledge to specify requirements, design, realize, verify, validate, operate and maintain a swarm intelligence system. The code-and-fix development method for swarm robotics systems relies completely on the ingenuity and experience of the developer. On the contrary, the proposed property-driven design offers several advantages.

First, property-driven design is an iterative process that guides and helps the designer in developing the system. The great majority of iterations occur when building the model. This allows the developer to focus only on the important aspects of the system because "whereas a simulation should include as much detail as possible, a good model should include as little as possible" [28],

Another advantage is that the risk of developing a system that does not satisfy the required properties is reduced, as these properties are evaluated at each step of the design and development phase. With the code-and-fix development these properties are either not verified or verified a posteriori, so the risk of developing the "wrong" system is high. Note that this holds only if the model is "good", meaning that it is able to faithfully represent the system. In this

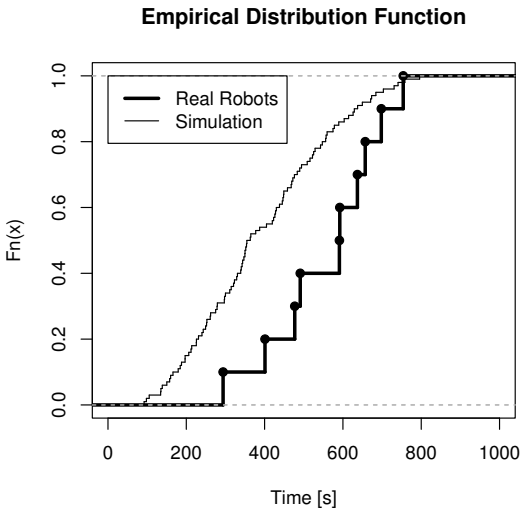


Figure 8: A graph showing the empirical cumulative distribution function ($Fn(x)$) of the results obtained with real robots (10 runs) and in simulation (100 runs). In both cases $p_{min-ac} = 0.22$.

paper, we did not discuss in details model validation. The interested reader can see, for example, Martinoli et al. [22].

Finally, since the developed model is hardware independent, it can be used to choose the robotic platform that fits best the characteristics of the system. Also, the model can be partially or completely reused for other applications, limiting the issue known as "reinventing the wheel". In the future, it is possible to imagine a set of publicly available models for swarm robotics applications that can be reused and modified by other developers.

While property-driven design has many advantages, it also has some limits. Being based on modeling, property-driven design inherits its advantages and limits. As with modeling, property-driven design can be applied to a large variety of swarm robotics systems. However, modeling a swarm robotics system is a hard task on its own. Many critical aspects of a swarm robotics system, such as robot-to-robot interaction or time and spatial aspects of the system are not always easy to capture in a model. Fortunately, many aspects of modeling a swarm robotics system have been studied extensively over the years (see, for instance, a review on modeling [21]).

Property-driven design can guide the developer in designing and developing a swarm robotics system. However, depending on the complexity of system to develop, implementing the model in simulation (phase 3) might be complicated. In these cases the ingenuity and expertise of the developer are still necessary.

7. CONCLUSION

In this paper, we presented property-driven design: a top-down design method based on the idea that a swarm robotics system can be described through a series of properties. Once these properties have been specified, it is possible to create a model of the system that satisfies them. In an iterative process, the model is improved until it correctly describes the system that the developer wants to design and satisfies the desired properties. The obtained model can then guide the development of a computer simulated version of the system.

Property-driven design is one of the first attempts towards the development of swarm engineering. Differently from code-and-fix development, property-driven design offers a systematic approach towards the development of a swarm robotics system. Among the advantages of property-driven design, compared with code-and-fix development, there are: a shift of the focus of the development process from implementation to design, given that most of the iterations happen at the design level; a formal way to specify the requirements of the system; a reduced risk of developing the "wrong" system, that is, a system that does not satisfy the requirements; and the possibility to develop a set of hardware independent models that can be reused for future applications.

In the future we plan to apply property-driven design to more complex applications, possibly using different modeling approaches. One problem to tackle is deriving the individual behavior of the robots starting from a collective behavior. Several possibilities can be studied, such as the integration of property-driven design with spatial computing or artificial evolution.

8. ACKNOWLEDGMENTS

This work was partially supported by the European Union through the ERC Advanced Grant "E-SWARM: Engineering Swarm Intelligence Systems" (contract 246939) and by the Future and Emerging Technologies project "ASCENS" (contract 257414).

Manuele Brambilla, Mauro Birattari and Marco Dorigo acknowledge support from the F.R.S.-FNRS of Belgium's Wallonia-Brussels Federation, of which they are a F.R.I.A. Research Fellow, a Research Associate and a Research Director, respectively.

9. REFERENCES

[1] R. Abbott. Emergence explained. *Complexity*, 12(1):13–26, 2006.

[2] J. Bachrach, J. Beal, and J. McLurkin. Composable continuous-space programs for robotic swarms. *Neural Computation & Applications*, 19:825–847, 2010.

[3] E. Bahçeci and E. Şahin. Evolving aggregation behaviors for swarm robotic systems: A systematic case study. In *Proceedings of the 2005 Swarm Intelligence Symposium - (SIS 2005)*, pages 333–340, Piscataway, NJ, 2005. IEEE Press.

[4] L. Bayindir and E. Şahin. A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering*, 15(2):115–147, 2007.

[5] K. Beck. *Test-driven Development: By Example*. Addison-Wesley, Boston, MA, 2003.

[6] S. Berman, A. Halasz, M. Hsieh, and V. Kumar. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, 25(4):927–937, 2009.

[7] B. Boehm. A spiral model of software development and enhancement. *Computer*, 21(5):61–72, 1988.

[8] M. Brambilla, C. Pinciroli, M. Birattari, and M. Dorigo. Property-driven design for swarm robotics:

Complete data, 2011. Supplementary information page at http://iridia.ulb.ac.be/supp/IridiaSupp2011-018/.

[9] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-Organization in Biological Systems*. Princeton Studies in Complexity. Princeton University Press, Princeton, NJ, 2001.

[10] F. Ciesinski and M. Größer. On probabilistic computation tree logic. In *Validation of Stochastic Systems*, number 2925 in Lecture Notes in Computer Science, pages 333–355. Springer, Berlin, Germany, 2004.

[11] E. Clarke. Model checking. In *Foundations of Software Technology and Theoretical Computer Science*, number 1346 in Lecture Notes in Computer Science, pages 54–56. Springer, Berlin, Heidelberg, 1997.

[12] D. Cleary. Perspectives on complex-system engineering. *Collaborations*, 3(2):1–4, 2005.

[13] E. Şahin. Swarm robotics: from sources of inspiration to domains of application. In *Swarm Robotics*, volume 3342 of *Lecture notes in computer science*, pages 10–20. Springer, Berlin, Heidelberg, 2005.

[14] C. Dixon, A. Winfield, and M. Fisher. Towards temporal verification of emergent behaviours in swarm robotic systems. In *Towards Autonomous Robotic Systems*, volume 6856 of *Lecture Notes in Computer Science*, pages 336–347. Springer, Berlin, Heidelberg, 2011.

[15] A. Gutiérrez, A. Campo, M. Dorigo, J. Donate, F. Monasterio-Huelin, and L. Magdalena. Open e-puck range & bearing miniaturized board for local communication in swarm robotics. In *IEEE International Conference on Robotics and Automation – ICRA 2009*, pages 3111–3116. IEEE Press, Piscataway, NJ, 2009.

[16] H. Hansson and B. Jonsson. A logic for reasoning about time and reliability. *Formal Aspects of Computing*, 6(5):512–535, 1994.

[17] A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. Prism: A tool for automatic verification of probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, number 3920 in Lecture Notes in Computer Science, pages 441–444. Springer, Berlin, Germany, 2006.

[18] R. Jeanson, C. Rivault, J.-L. Deneubourg, S. Blanco, R. Fournier, C. Jost, and G. Theraulaz. Self-organized aggregation in cockroaches. *Animal Behaviour*, 69(1):169–180, 2005.

[19] S. Kazadi, J. R. Lee, and J. Lee. Model independence in swarm robotics. *International Journal of Intelligent Computing and Cybernetics, Special Issue on Swarm Robotics*, 2(4):672–694, 2009.

[20] S. Konur and C. Dixon. Formal verification of probabilistic swarm behaviours. In *Swarm Intelligence, 7th International Conference, ANTS 2010*, volume 6234 of *Lecture Notes in Computer Science*, pages 572–573. Springer, Berlin, Germany, 2010.

[21] K. Lerman, A. Martinoli, and A. Galstyan. A review of probabilistic macroscopic models for swarm robotic systems. In *Swarm robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 143–152. Springer, Berlin, Heidelberg, 2005.

[22] A. Martinoli, A. J. Ijspeert, and F. Mondada. Understanding collective aggregation mechanisms: from probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems*, 29(1):51–63, 1999.

[23] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, volume 1, pages 59–65, Portugal, 2009. IPCB: Instituto Politécnico de Castelo Branco.

[24] S. Nolfi and D. Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA, 2000.

[25] R. O'Grady, C. Pinciroli, A. L. Christensen, and M. Dorigo. Supervised group size regulation in a heterogeneous robotic swarm. In *Proceedings of ROBOTICA 2009 - 9th International Conference on Autonomous Robot Systems and Competitions*, pages 113–119. IPCB, Castelo Branco, Portugal, 2009.

[26] L. Panait and S. Luke. Cooperative multi-agent learning: the state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[27] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. A. Di Caro, F. Ducatelle, T. Stirling, A. Gutiérrez, L. M. Gambardella, and M. Dorigo. ARGoS: a modular, multi-engine simulator for heterogeneous swarm robotics. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'11)*, pages 5027–5034. IEEE Computer Society Press, Los Alamitos, CA, 2011.

[28] J. M. Smith. *Models in ecology*. Cambridge University Press, Cambridge, MA, 1978.

[29] O. Soysal and E. Şahin. A macroscopic model for self-organized aggregation in swarm robotic systems. In *Swarm Robotics*, volume 4433 of *Lecture Notes in Computer Science*, pages 27–42. Springer, Berlin, Heidelberg, 2007.

[30] W. M. Spears, D. F. Spears, J. C. Hamann, and R. Heil. Distributed, physics-based control of swarms of vehicles. *Autonomous Robots*, 17(2–3):137–162, 2004.

[31] S. Stepney, F. Polack, and H. R. Turner. Engineering emergence. In *11th IEEE International Conference on Engineering of Complex Computer Systems, 2006. ICECCS 2006*, pages 89–97. IEEE Press, Piscataway, NJ, 2006.

[32] V. Trianni, R. Groß, T. H. Labella, E. Şahin, and M. Dorigo. Evolving aggregation behaviors in a swarm of robots. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, editors, *Proceedings of the Seventh European Conference on Artificial Life (ECAL 2003)*, volume 2801 of *Lecture Notes in Computer Science*, pages 865–874. Springer, Berlin, Heidelberg, 2003.

Multi-robot collision avoidance with localization uncertainty

Daniel Hennes
Maastricht University
P.O. Box 616, 6200MD
Maastricht, The Netherlands
daniel.hennes@gmail.com

Daniel Claes
Maastricht University
P.O. Box 616, 6200MD
Maastricht, The Netherlands
danielclaes@me.com

Wim Meeussen
Willow Garage
68 Willow Rd., Menlo Park
CA 94025, USA
meeussen@willowgarage.com

Karl Tuyls
Maastricht University
P.O. Box 616, 6200MD
Maastricht, The Netherlands
k.tuyls@maastrichtuniversity.nl

ABSTRACT

This paper describes a multi-robot collision avoidance system based on the velocity obstacle paradigm. In contrast to previous approaches, we alleviate the strong requirement for perfect sensing (i.e. global positioning) using Adaptive Monte-Carlo Localization on a per-agent level. While such methods as Optimal Reciprocal Collision Avoidance guarantee local collision-free motion for a large number of robots, given perfect knowledge of positions and speeds, a realistic implementation requires further extensions to deal with inaccurate localization and message passing delays. The presented algorithm bounds the error introduced by localization and combines the computation for collision-free motion with localization uncertainty. We provide an open source implementation using the Robot Operating System (ROS). The system is tested and evaluated with up to eight robots in simulation and on four differential drive robots in a real-world situation.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms, Experimentation

Keywords

multi-robot systems, optimal reciprocal collision avoidance, adaptive monte-carlo localization, robot operating system

1. INTRODUCTION

Local collision avoidance is the task of steering free of collisions with static and dynamic obstacles, while following a global plan to navigate towards a goal location. Figure 1 shows a configuration of two robots on collision course. The task of the algorithm is to avoid collision with a minimal deviation of the preferred path. Thus, local collision avoidance differs from motion planning, global path planning and local path planning. In motion planning the environment of the robot is assumed to be deterministic and known in advance, thus allowing to plan a complete path to the goal. Global path planners usually operate on a static map and find either the minimum cost plan (e.g. using A* or Dijkstra's algorithm) or any valid plan (e.g. sample based planners). Local path planners, such as Trajectory Rollout and Dynamic Window Approaches (DWA), perform forward simulations for a set of velocity commands; each resulting trajectory is scored based on proximity to the goal location and a cost map built from current sensor data. In principle this allows to stay clear of dynamical obstacles; however, in multi-robot settings two problems arise:

1. Robots are not merely dynamic obstacles; each robot itself is a pro-active agent taking actions to avoid collisions. Neglecting this might lead to *oscillations* and thus highly inefficient trajectories or even collisions.

2. The sensor source (e.g. laser range finder) is usually mounted on top of the robot's base to allow for a maximal unoccluded viewing angle. In a system with homogenous robots this implies that there is very little surface area that can be picked up by the sensors of other robots and thus prevents the robots from observing each other.

Local collision avoidance addresses these challenges and is an important building block in any robot navigation system targeted at multi-robot systems. Although robot localization is a requirement for collision avoidance, most approaches assume perfect sensing and positioning and avoid local methods by using global positioning via an overhead tracking camera - or are purely simulation based. Nevertheless, to be able to correctly perform local collision avoidance in a realistic environment, a robot needs a reliable position estimation without the help of external tools.

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

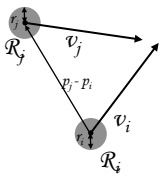


Figure 1: A workspace configuration with two robots R_i and R_j on collision course.

Our approach uses Optimal Reciprocal Collision Avoidance (ORCA) and the extension to non-holonomic robots (NH-ORCA) [1] in combination with Adaptive Monte-Carlo Localization (AMCL) [3].This effectively alleviates the need for global positioning by decentralized localization on a per-agent level. We provide a solution that is situated in between centralized motion planning for multi-robot systems and communication-free individual navigation. While actions will remain to be computed independently for each robot, information about position and velocity is shared using local inter-robot communication. This keeps the communication overhead limited while avoiding problems like robot-robot detection. The resulting algorithm is implemented in the open source Robot Operating System (ROS), which provides hardware abstraction and message-passing. Our experiments in simulation and on a physical system illustrate the feasibility and efficiency of the approach.

The remainder of the paper is structured as follows. Section 2 provides background information about ORCA, NH-ORCA, AMCL and ROS. Section 3 discusses key challenges in applying velocity-based collision avoidance to real-world robotic scenarios, leading to the proposed approach. In Section 4, we introduce our novel method to incorporate localization uncertainty. Experimental results are presented in Section 5. The paper concludes with a brief discussion and highlights future directions of this work in Section 6.

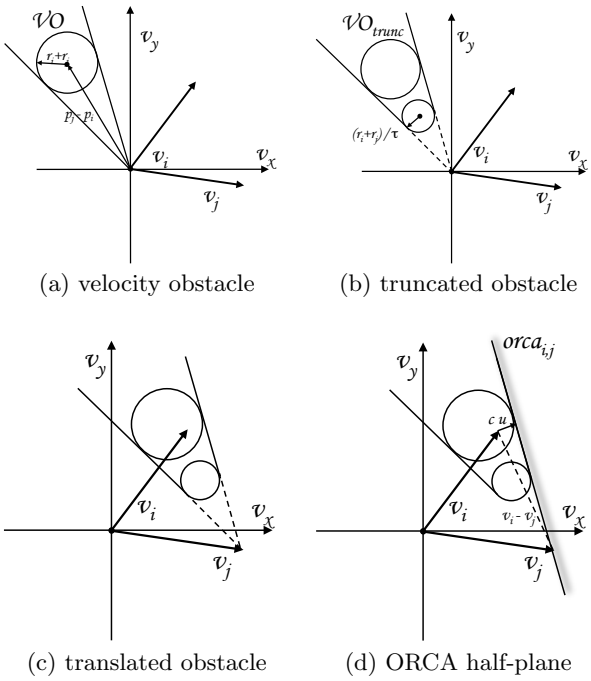
2. BACKGROUND

In this section, we will concisely describe the collision avoidance algorithms ORCA and NH-ORCA and the Adaptive Monte Carlo Localization (AMCL) method. Additionally, the Robot Operating System (ROS) will be introduced.

2.1 Optimal Reciprocal Collision Avoidance

Our work is based on the principle of *Optimal Reciprocal Collision Avoidance (ORCA)* introduced by van den Berg et al. [10], an extension of Reciprocal Velocity Obstacles (RVO) [11]. ORCA is a velocity-based approach [2] to achieve collision avoidance in multi-agent systems taking into account the motions of other agents. For simplicity, the two dimensional case is assumed, but the formulations can be extended into the third dimension.

ORCA describes a control policy were each agent selects a collision-free velocity from the two dimensional *velocity space* in x and y direction. This implies that a holonomic robot is assumed in the original formulation, since the robot has to be able to accelerate into every direction regardless of its current state. However, the movement model of a non-holonomic robot can be incorporated by limiting the velocity space accordingly, see Section 2.2.



(a) velocity obstacle (b) truncated obstacle

(c) translated obstacle (d) ORCA half-plane

Figure 2: Creating velocity obstacles and ORCA half-planes based on the workspace configuration shown in Figure 1. (a) Translating the situation into velocity space and the resulting velocity obstacle for R_i, assuming a static obstacle. (b) Truncating the velocity obstacle (VO) for time frame τ. Every velocity that is allowed will be collision-free for at least time τ. (c) Translating the VO according to the other robot's velocity v_j. Velocity v_i points into the translated VO, hence R_i is on collision course. (d) Creating an ORCA half plane for R_i. Vector u is the minimal vector to add to v_i to be outside of the VO. The perpendicular line to u at v_i+cu is the set of velocities that is closest to v_i and avoiding collisions for $c \geq 1$. For $c = \frac{1}{2}$ the robot R_i assumes that R_j will take care of the other half of the collision avoidance.

Let us assume a workspace configuration with two robots on a collision course as shown in Figure 1. If the position of agent R_j is known to R_i, a region in the robot's velocity space can be calculated which is leading to a collision under current velocities and is thus unsafe. If we assume disc-shaped robots, which are not moving, the region of non-allowed velocities is bounded by the half-lines emanating from the origin, tangent to a disk at the relative position of the two agents with the combined radius of the two robots as in Figure 2(a). These unsafe regions are called velocity obstacles (VO). Taking a velocity in direction of the other agent will not immediately result in a collision, hence the VO can be bounded by a given time frame τ, leading to a truncated cone. Taking a velocity which is now available again will be collision free for at least the given time frame, see Figure 2(b). If we now assume that both agents are moving, the VO has to be translated into the direction the speed of R_j as shown in Figure 2(c). Now, robot R_i's velocity vector v_i points into the VO, thus we know that R_i and R_j are on collision course. Each agent computes a VO for each of the other agents. If all agents at any given time step select

velocities outside of the VOs, the trajectories are guaranteed to be collision free. However, oscillations can still occur [11].

To overcome the problem of oscillations and to enable efficient calculation for safe velocities, *Optimal Reciprocal Collision Avoidance (ORCA)* was introduced by van den Berg et al. [10]. Instead of velocity obstacles, agents independently compute half-planes of collision-free velocities for each other agent as shown in Figure 2(d). The half planes are selected to be as close to the desired goal velocity as possible. Thus, they are parallel lines to either one of the two legs of the VO. If we assume reciprocal collision avoidance, the line can be slightly insight the VO, assuming that the other robot will take care of the other half of the collision avoidance. The intersection of all half planes is the set of collision free velocities. The optimal velocity from this set can be calculated by solving a linear program minimizing the distance to the desired goal velocity.

Though each agent selects a new velocity independently, a distributed implementation of ORCA on a physical system of mobile robots requires *perfect sensing* of the shape, position and velocities of other robots. A variant of the original RVO called *Hybrid Reciprocal Velocity Obstacles (HRVO)* is presented in [7]. This extension takes uncertainty of movement and sensing into account; however, it uses global positioning via an overhead camera and does not incorporate the ORCA formulation.

2.2 Kinematic constraints

As mentioned above, the original formulation of ORCA is based on holonomic robots, which can accelerate into any direction from every state. However, differential drive robots with only two motorized wheels are much more common due to their lower price point. To incorporate the differential drive constraints, Kluge et al. introduced a method to calculate the effective center of a differential drive robot [5]. The effective center represents a translation of the center of rotation to a point that can virtually move into all directions. It can be incorporated in the ORCA formulation by virtually enlarging the robots' radii prior to the calculations. These adaptations provide additional maneuverability to handle the differential drive constraints. ORCA-DD [8] extends this idea and enlarges the robot to twice the radius of the original size to ensure collision free and smooth paths for robots under differential constraints. The effective center is then located on the circumference of the robot at the center of the extended radius. However, this quadruples the virtual size of the robot, which can result in problems in narrow corridors or unstructured environments.

Another method to handle non-holonomic robot kinematics has been introduced by Alonso-Mora et al [1]: NH-ORCA is the generalized version of ORCA for any non-holonomic robot. The underlying idea is that any robot can track a holonomic speed vector with a certain tracking error ε. This error depends on the direction and length of the holonomic velocity, i.e. a differential drive robot can drive along an arc and then along a straight line which is close to a holonomic vector in that direction. A set of allowed holonomic velocities is calculated based on the current speed and a maximum tracking error ε. Resulting constraints are added to the linear program in the ORCA formulation. To allow smooth and collision free navigation, the virtual robot radii have to be increased by the tracking error ε, since the robots do not track the desired holonomic velocity exactly. Additionally, in dense configuration with many robots, turn in-place can be included by adapting the allowed tracking error ε dynamically depending on the current state (i.e. proximitiy to other robots and current velocities). The set of allowed holonomic velocities can be calculated for any possible angle and error. However, any further constraint in the linear program slows down the computation, thus the feasible set can be approximated by a polygon.

NH-ORCA is preferred over ORCA-DD, since the virtual increase of the robots' radii is only by a size of ε instead of doubling the radii.

2.3 Adaptive Monte-Carlo Localization

The localization method employed in our work is based on sampling and importance based resampling of particles, in which each particle represents a possible pose and orientation of the robot. More specifically, we use the adaptive monte-carlo localization method, which dynamically adapts the number of particles [3]. Monte-Carlo Localization (also known as a particle filter), is a widely applied localization method in the field of mobile robotics. It can be generalized in an initialization phase and two iteratively repeated subsequent phases, the prediction and the update phase.

In the initialization phase, a particle filter generates a number of samples N, which are uniformly distributed over the whole map of possible positions. In the 2.5D case, every particle s^i has a x- and y-value and a rotation $s^i = (\hat{x}, \hat{y}, \hat{\theta})$. The particles are usually initialized in such a way, that only valid positions are taken into account, i.e. they cannot be outside of the map or within walls.

The first iterative step is the prediction phase, in which the particles of the previous population are moved based on the motion model of the robot, i.e. the odometry. Afterwards, in the update phase, the particles are weighted according to the likelihood of the robot's measurement for each particle. Given this weighted set of particles the new population is resampled in such a way that the new samples are selected according to the weighted distribution of particles in the old population. In the following, the two phases are explained in further detail.

Prediction phase: After each movement, the position of each particle is updated according to the belief of the agent. More specifically, if the robot has moved forward 10 cm, each particle is moved 10 cm into the direction of its rotation. If the robot rotates, the particles are rotated accordingly. Thus, if a holonomic robot moves from state $\mathbf{x_k} = (x_k, y_k, \theta_k)$ to $\mathbf{x_{k+1}} = (x_{k+1}, y_{k+1}, \theta_{k+1})$, the particles are translated by:

$$\begin{bmatrix} \hat{x}_{k+1} \\ \hat{y}_{k+1} \\ \hat{\theta}_{k+1} \end{bmatrix} = \begin{bmatrix} \hat{x}_k + \rho cos(\hat{\theta}_k + \Delta\theta) \\ \hat{y}_k + \rho sin(\hat{\theta}_k + \Delta\theta) \\ \hat{\theta}_k + \Delta\theta \end{bmatrix} \quad (1)$$

Where $\rho = \sqrt{\Delta x^2 + \Delta y^2}$ and $\Delta\theta = \theta_k - \theta_{k+1}$. However, both ρ and $\Delta\theta$ are corrupted by noise due to errors in actuators and odometry. Hence, the more accurate the robot's motion model is, the better the performance of the prediction phase. For non-holonomic robots, the update equations can be changed accordingly [9].

Update phase: After a sensor update, the expected measurement for each particle is calculated. This means, measured sensor values are compared with the world view that

is expected if the robot would be at the position of the particle (i.e. by a laser scan matcher). The new weight (w_k^i) is the probability of the actual sensor measurement (z_k) given the particles position (s_k^i) at time k as shown below:

$$w_{k+1}^i = p(z_k|s_k^i) \tag{2}$$

As w is a probability distribution, the weight for each particle is re-normalized after each update:

$$w_k^i = \frac{w_k^i}{\sum_i w_k^i} \tag{3}$$

The resampling can be done in linear time as described in [9]. After resampling, all the weights are reset to the uniform weight of $1/N$.

Resampling and variable sample set size: Particle filters only need a large N to correctly identify the position when the initial state is unknown. However, when the present localization is quite accurate already, less particles are needed to keep track of the position changes. Hence, the number of samples can be changed adaptively depending on the position uncertainty. We use the approach of KLD-sampling, which determines the minimum number of samples needed, such that with probability $1 - \delta$ the error between the true posterior and the sample-based approximation is less than ε. The number of samples can be calculated as:

$$n = \frac{1}{2\varepsilon}\chi^2_{k-1,1-\delta} \tag{4}$$

This can be approximated using the Wilson-Hilferty transformation as:

$$n = \frac{k-1}{2\varepsilon}\left\{1 - \frac{2}{9(k-1)} + \sqrt{\frac{2}{9(k-1)}}z_{1-\delta}\right\} \tag{5}$$

where k is the number of bins of the discrete distribution from which the particles are sampled. For further details we refer to [4].

Kidnapped robot and false localization: A common problem occurs if there are several locations which are represented similar according to the sensor values. For example two hallways, which have only one door on the right. In these cases, it happens that the robot localizes itself at the wrong position. Furthermore, the robot can be moved by an external force, like a human. To incorporate sudden changes or wrong localizations, a fraction of particles can be moved to a random location. This increases the robustness of the system.

In our work, AMCL is not used for global localization, but rather initialized with a location guess that is within the vicinity of the true position. This enables us to use AMCL for an accurate position tracking without having multiple possible clusters in ambiguous cases.

2.4 Robot Operating System (ROS)

The NH-ORCA and the AMCL algorithms are implemented in the framework of the open source *Robot Operating System (ROS)* [6]. ROS provides many useful tools, hardware abstraction and a message passing system between nodes. Nodes are self contained modules that run independently and communicate which each other over so called *topics* using a one-to-many subscriber model and the TCP/IP protocol. Naturally this is of great importance when working with distributed systems. In addition, the modularity enables to easily create various configurations for different settings; to run our system on ROS-enabled robots, only the parameters need to be adapted according to the robot's motion and sensor model.[1].

3. PROBLEM DESCRIPTION AND APPROACH

We propose a system that builds upon the two main components introduced in Section 2, i.e. NH-ORCA and AMCL, to provide collision free motion in a real-world system of robots. In this section we will revisit the assumptions commonly made by all velocity-based collision avoidance algorithms and motivate our choice for per agent-based localization in combination with position and velocity information sharing using inter-robot communication. Furthermore, we will point out the necessary addition of sensor uncertainty, leading to our proposed algorithm *Collision Avoidance with Localization Uncertainty (CALU)* explained in more detail in Section 4.

3.1 Problem description

ORCA (and all its variants) does not require any inter-robot negotiation to find optimal collision free motion trajectories and is hence in principal fully distributed. However, all methods require perfect information about the positions, velocities and shapes of all other robots. In order to preserve the distributed nature of this approach, robots need to be able to accurately identify other robots using on-board sensors; furthermore, positions and velocities have to be deduced from the same data. The list of typical sensors for mobile robots includes stereo cameras, laser range finders and lately 3D image sensors (e.g. Microsoft Kinect). These sensors deliver large data-streams that require considerable computational power to process even for the detection and classification of static obstacles.

The computational requirement is not the only problem when considering robot-robot detection. As low-end laser range finders (e.g. Hokuyo URG-04LX) become widely available even for mobile robotic projects on a small budget, they are the preferred sensor choice due to their high accuracy, resolution and field of view. However, the laser range finder is usually mounted on top of the robot's base to allow for a maximal unoccluded viewing angle. In a system with homogenous robots that means that there is very little surface area that can be picked up by the sensors of other robots and thus prevents the robots from observing each other.

Even though the laser range finder provides a high accuracy in the readings, the localization and tracking of the robot using AMCL will in general have the tendency to differ to some extent from the true position of the robot. If the size of the localization and tracking error is in the order of magnitude of the robots radius, collisions are bound to happen.

Previous approaches have worked around these problems by providing global positioning to all robots based on an overhead tracking camera. Such a system is not distributed since a host computer connected to the camera needs to process the sensor data and communicate with all robots to

[1]For more information see: `http://www.ros.org/`.

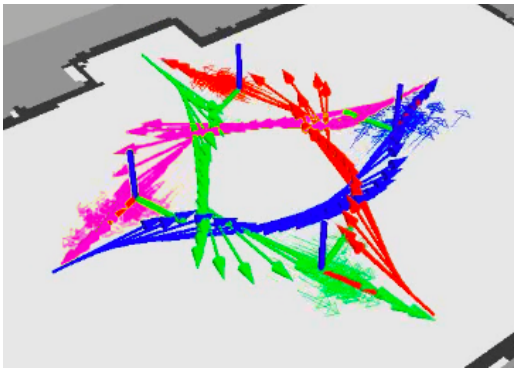


Figure 3: CALU with four robots. ROS visualization tool RVIZ is used to show the trajectories and localization particles of four robots.

provide position and velocity data. If this machine fails the system breaks.

3.2 Approach

We propose to utilize agent-based localization and inter-robot communication to provide a system that is more realistic in real-world scenarios (i.e. without the need for external positioning data) and also more robust (i.e. single component failure does not lead to system failure). Our approach, called Collision Avoidance with Localization Uncertainty (CALU), results in a fully decentralized system that uses local communication to share robot state information in order to ensure smooth collision free motion; an example for 4 robots is shown in Figure 3. Below we describe the four key components of this approach.

Platform: The robots are assumed to be differential drive robots. Required sensors are a laser range finder and wheel odometry. For simplicity we assume a circular footprint; other shapes can be approximated by the circumscribed radius. In order to connect the different subsystems, including device drivers and software modules, we use ROS (see Section 2.4).

Sensor processing and localization: Each robot integrates wheel odometry data which is in turn used to drive the motion model of AMCL (see Section 2.3), hence tracking the pose of the robot. Laser range finder scans are used in the update phase of AMCL. The uncertainty of the current localization, i.e. the spread and weight of the particles, is taken into account for the calculation of collision free velocities as will be explained in further detail in Section 4. We assume a prior static map that is used for localization and available to all robots, thus providing a consistent global coordinate frame.

Inter-robot communication: Each robot broadcasts its position and velocity information in the global coordinate frame on a common ROS topic. Each robot also subscribes to the same topic and caches position and velocity data of all other robots. Message delays are taken into account and positions are forward integrated in time according to the motion model of robots using the last known position and velocity information.

Collision avoidance: NH-ORCA (see Section 2.2) is used to compute optimal collision free velocities according to the aggregated position and velocity data of all surrounding robots. As a last step we incorporate localization uncertainty in the NH-ORCA computation as detailed in Section 4. The allowed tracking error is scaled depending on current speed of the robot.

4. LOCALIZATION UNCERTAINTY

The key idea of CALU is to bound the error introduced by localization. To derive this bound, we revisit the particle filter described in Section 2.3.

Let $\mathbf{x}_k = (x, y, \theta)$ be the state of the system. The posterior filtered density distribution $p(\mathbf{x}_k|\mathbf{z}_{1:k})$ can be approximated as:

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N} w_k^i \; \delta\left(\mathbf{x}_k - \mathbf{s}_k^i\right) \tag{6}$$

where $\delta(\cdot)$ is the Dirac delta measure. We recall that a particle state at time k is captured by $\mathbf{s}_k^i = (\hat{x}_k^i, \hat{y}_k^i, \hat{\theta}_k^i)$. In the limit $N \to \infty$, Equation 6 approaches the real posterior density distribution. We can define the mean $\mu = (\mu_x, \mu_y, \mu_\theta)$ of the distribution accordingly:

$$\mu_x = \sum_i w_k^i \; \hat{x}_k^i \tag{7}$$

$$\mu_y = \sum_i w_k^i \; \hat{y}_k^i \tag{8}$$

$$\mu_\theta = atan2\left(\sum_i w_k^i \; sin(\hat{\theta}_k^i), \sum_i w_k^i \; cos(\hat{\theta}_k^i)\right) \tag{9}$$

The mean gives the current position estimate of the robot. However, the estimate is likely to be noisy and we have to take this uncertainty into account in order to ensure collision free motion. The probability of the robot residing within a certain area $\mathcal{A}$ at time k is:

$$p(\mathbf{x}_k \in \mathcal{A}|\mathbf{z_{1:k}}) = \int_{\mathcal{A}} p(\mathbf{x}|\mathbf{z_{1:k}}) d\mathbf{x} \tag{10}$$

We can rewrite (10) using (6) as follows:

$$p(\mathbf{x}_k \in \mathcal{A}|\mathbf{z_{1:k}}) \approx \sum_{\forall_i : \mathbf{s}_k^i \in \mathcal{A}} w_k^i \; \delta\left(\mathbf{x}_k - \mathbf{s}_k^i\right) \tag{11}$$

From (11) we see that for any given $\varepsilon \in [0, 1)$ there is an $\mathcal{A}$ such that:

$$p(\mathbf{x}_k \in \mathcal{A}|\mathbf{z_{1:k}}) \geq 1 - \varepsilon \tag{12}$$

Given sufficient samples, the localization uncertainty is thus bounded and we can guarantee that the robot is located within area $\mathcal{A}$ with probability $1 - \varepsilon$.

ORCA as well as NH-ORCA assume disc-shaped robots to make calculations tractable. If a robot radius is inflated by d, the center point of the robot can in turn be translated by a maximum distance of d from its original position while the resulting disc still circumscribes the entire robot. We next derive d such that (12) holds.

We define a subset $\mathbf{S} \subset \{\mathbf{s}^1, \ldots, \mathbf{s}^N\}$ with

$$d_{\mathbf{S}} = \max_{(x,y,\theta) \in \mathbf{S}} \left((x - \mu_x)^2 + (y - \mu_y)^2\right) \tag{13}$$

the maximal distance to the mean. Furthermore, we define:

$$\mathcal{S} : \mathbf{S} \in \mathcal{S} \text{ iff } p(\mathbf{x}_k \in \mathbf{S}|\mathbf{z_{1:k}}) \geq 1 - \varepsilon$$

# of robots	GT	NH-ORCA $\sigma = 0.0m$	NH-ORCA $\sigma = 0.2m$	CALU $\sigma = 0.0m$	CALU $\sigma = 0.2m$
2	0	0	0	0	0
4	0	0	62 (11)	0	0
6	0	7 (4)	85 (11)	0	0
8	0	17 (5)	391 (35)	0	1 (1)
loc. error	–	0.064 $\pm$ 0.009	0.127 $\pm$ 0.028	0.061 $\pm$ 0.011	0.117 $\pm$ 0.043

Table 1: Resulting collisions with various settings summed over 50 runs. The number in brackets shows the number of runs in which the collisions occurred. AMCL is either initialized with a perfect guess or initial guesses sampled from a two dimensional normal distribution ($\sigma_x = \sigma_y = 0.2m$) centered around the ground truth position.

There is a minimal subset $\mathbf{S}^* \in \mathcal{S}$ such that (12) holds and the maximal distance to the mean is minimized:

$$\mathbf{S}^* = \arg\min_{\mathbf{S}\in\mathcal{S}} d_{\mathbf{S}} \tag{14}$$

Thus, if the robot radius is inflated by $d = d_{\mathbf{S}^*}$ the resulting disc circumscribes the entire robot with a probability of $1-\varepsilon$.

The implementation of this computation is straightforward and efficient. An implementation of AMCL as explained in Section 2.3 commonly tracks particles in a k-d tree structure. The algorithm localizes the node closest to the mean μ and subsequently increases the radius d while adding particles that fall into the radius to the set $\mathbf{S}^*$ and accumulating the weight sum until the threshold $1 - \varepsilon$ is reached.

5. EXPERIMENTS AND RESULTS

This section presents experiments and results of the proposed system. We have evaluated our approach in simulation using *Stage* [12] and in a real-world setting.

5.1 Simulation experiments

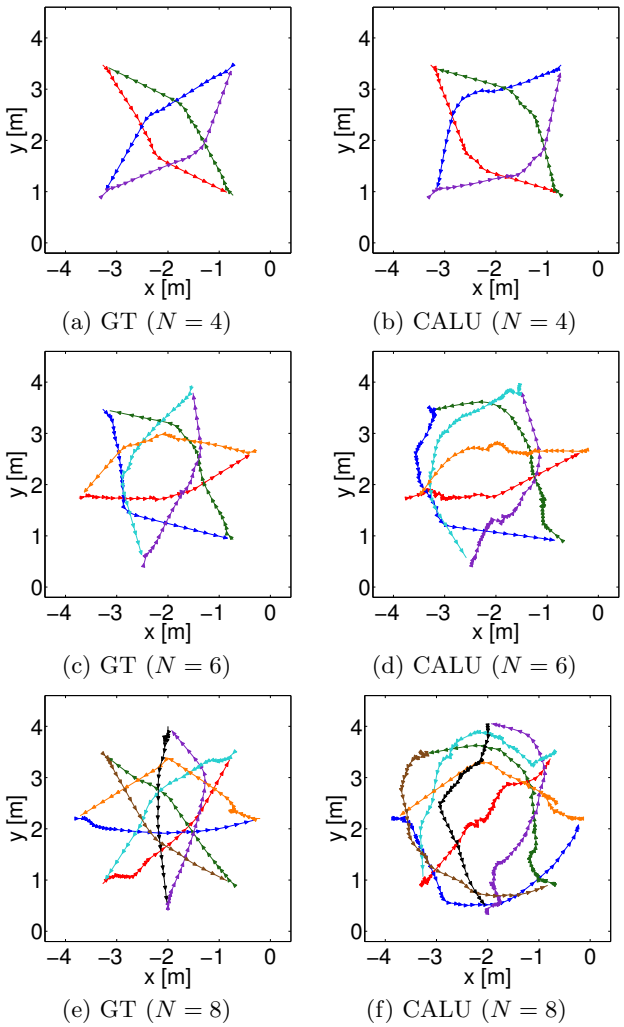
Simulation allows us to investigate the system performance when using localization in comparison to ground truth positioning with perfect information. For evaluation we have chosen seven different scenarios, using two to eight robots. In each setting, the robots where located on a circle (equally spaced) with a radius of 1.8 meter and the goals located on the antipodal positions, i.e. each robot's shortest path is through the center of the circle. The goal is assumed to be reached, when the robots center is within a 0.1 meter radius of the true goal.

Configurations: Each scenario is tested with three different configurations for localization:

Ground Truth (GT): Each robot gets perfect position and velocity information through the simulation environment.

AMCL with $\sigma = 0.0m$: Each robot starts AMCL initialized with the exact pose. The pose cloud is initialized with gaussian noise in x and y direction with $\sigma = 0.0m$.

AMCL with $\sigma = 0.2m$: Each robot starts AMCL initialized with initial guesses sampled from a 2-dimensional normal distribution ($\sigma_x = \sigma_y = 0.2m$) centered around the ground truth position.



(a) GT ($N = 4$) (b) CALU ($N = 4$)
(c) GT ($N = 6$) (d) CALU ($N = 6$)
(e) GT ($N = 8$) (f) CALU ($N = 8$)

Figure 4: Typical robot trajectories observed for different numbers of robots when comparing ground truth (GT) to CALU.

All three settings were tested using NH-ORCA and CALU for collision avoidance. When using ground truth both algorithms are essentially the same leading to a total of five different configurations.

Performance indices: We measure several performance indices: a) number of collisions, b) time to complete run, c) distance travelled, d) localization error and e) jerk cost. The jerk cost measures the smoothness of a path and is defined as:

$$Jerk_{lin} = \frac{1}{2}\int \dddot{\mathbf{x}}(t)\ dt\ , \qquad Jerk_{ang} = \frac{1}{2}\int \dddot{\theta}(t)\ dt\ ,$$

where $\mathbf{x}$ is the two dimensional position vector and θ the robot's heading.

System: Experiments were run on a single machine with a quad core 3.07 GHz Intel i7 processor and 6GB of memory. Each setting was repeated 50 times and results were averaged.

The results of the simulation experiments are summarized in Table 1. We can observe that the number of collisions using the original NH-ORCA rises immensely depending on

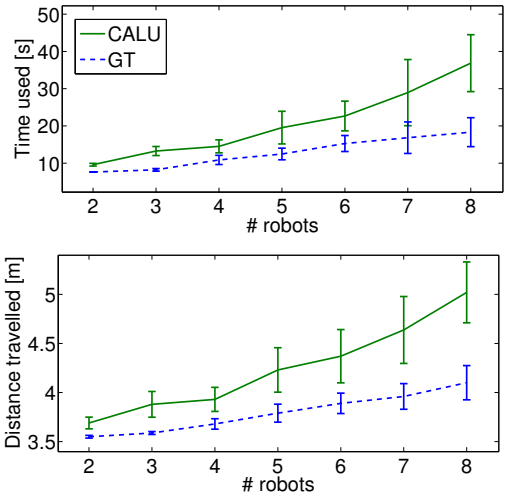


Figure 5: Total time (top) and distance travelled (bottom) metric for CALU and ground truth (GT) averaged over 50 simulations runs.

the localization error and the number of robots. Up to a total of 391 collisions in 35 runs. CALU stayed collision free; except for a single run, in which only one collision occurred. Therefore, for the further discussion NH-ORCA is excluded since it can not be seen as a realistic obstacle avoidance method without our adaptations. This leads us to further compare CALU and GT. To stay as realistic as possible, we focus on runs with CALU and an initial guess corrupted by gaussian noise. In reality, this will also be the case, since it is almost impossible to determine the real position of a robot on a map. Thus, when speaking of CALU in the coming paragraphs, it is referring to CALU with a noisy initial guess.

Some typical trajectories with ground truth (GT) and CALU that we observed during the simulation runs are presented in Figure 4. For up to seven robots the resulting trajectories are usually smooth. In the setting with eight robots, the relatively small area gets very crowded and there is hardly any space to maneuver, see Figure 4(f). Likewise, we can observe that using CALU generally results in larger arcs that are farther away than when using GT. This can be explained by the inflated radius when using CALU due to the sensor uncertainty.

As expected, the runtime and distance travelled increased for more robots as presented in Figure 5. CALU generally uses more time and travels longer than GT. This is also reflected in the average jerk costs, see Figure 6. Interestingly, CALU uses a lot more angular jerk than GT already for two and three robots, while GT increases a lot at first and then stabilizes below the CALU amounts. We assume that the large jerk costs already for only a few robots are due to the inflated radii based on the localization uncertainty. Especially right after initialization, the localization uncertainty is very large, since the particles are scattered more widely thus inflating the radius by a larger factor. This only stabilizes after a couple of update steps using the feedback from the odometry and the laser measurements.

5.2 Real-world experiments

In addition to simulation runs, we have investigated the performance of CALU in real-word settings up to four differential drive Turtlebots[2]. The robots are based on the iRobots Create platform and have a diameter of 33.5 cm. In addition to the usual sensors, they are equipped with a Hokuyo URG laser-range finder to enable better localization in large spaces. All computation is performed on-board on a Intel Atom D525 1.8GHz dual core CPU netbook. Communication between the robots is realized via a 2.4 GHz WiFi link. Before set up the robots are driven remotely to their initial positions and AMCL is initialized with an approximated initial guess.

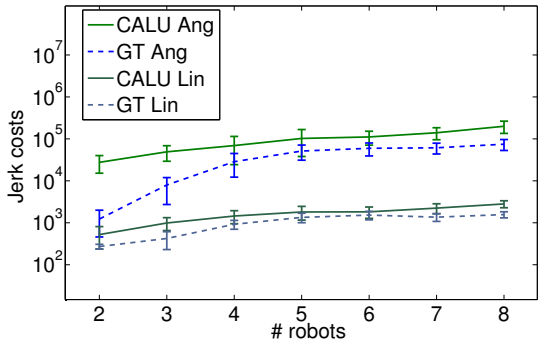


Figure 6: Linear and angular jerk costs for ground truth (GT) and CALU averaged over 50 simulation runs.

Figure 7 shows the trajectories of an example run of the four robots using CALU. The initial positions are approximately 3.5 meters apart; the goal location are set to the diagonally opposing start locations. The system successfully avoids collision and produces smooth paths; except for a small jump in the localization that can be observed in the path of robot starting in the upper right corner.

Additionally, we tested a realistic setting of two robots in a narrow hallway. Each robot wants to get to the other side of the hallway; thus having to pass the other robot. Figure 8 shows the setup and the resulting paths using CALU. To overcome that the robots drive into the walls, two ORCA lines where added to the robots. Adding these additional lines automatically, based on the map and sensor data, is topic of future work as described in the next section. The resulting paths are very close, but still collision free.

6. CONCLUSIONS

While the proposed approach works well in many cases, there are some limitations that we need to address. If the workspace gets more and more crowded with multiple robots, the resulting paths are not always smooth. (NH-)ORCA computes an optimal velocity that is collision free and closest to the desired velocity. However, in our experiments the desired velocity points always straight to the goal. Without a planning algorithm that plans multiple waypoints around fixed obstacles (and motionless robots) there can occur situations where the resulting velocity would be zero.

Differential drive robots can not follow the ORCA velocities directly and always have to turn on the spot or track an arc to accomplish the desired change of direction. In crowded situations, robots do have to turn on the spot in order to avoid collisions, while in open space collision avoid-

[2]For more information see: `http://turtlebot.com`.

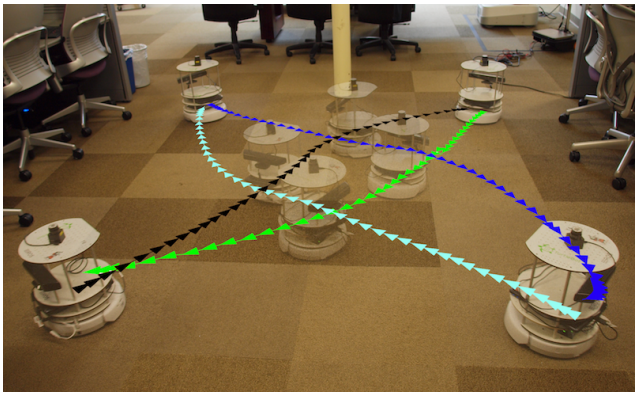


Figure 7: Real-world collision avoidance with four differential drive robots using CALU.

ance between only a few robots smoother arc tracking is desired. This is accomplished by the dynamic adaptation of the error bound depending on the speed.

If the size of the localization error is in the magnitude of the robot radius, collisions are bound to happen and this is shown in the results with NH-ORCA. This is resolved by introducing CALU and adapting the robots radii according to the localization uncertainty reported by AMCL. However, the combination of the dynamically adapting error bound and the inflation due to uncertainty might lead to oscillations, since previously free path become blocked and free again depending on the other agents' radii. To overcome this the radius scaling can be filtered. Additionally, the AMCL localization can jump and combined with delays in communication this can lead to collisions. An Extended Kalman Filter could be a possible solution to this problem.

In future work we will investigate these idea and furthermore extend our experiments to different scenarios, i.e. larger map, various start and goal location configurations and uncontrolled moving obstacles like humans. Additionally, we will examine the possibility of how to implement the presented algorithm as part of a global and local planner that will take the map and static obstacles obtained from the sensor data into account.

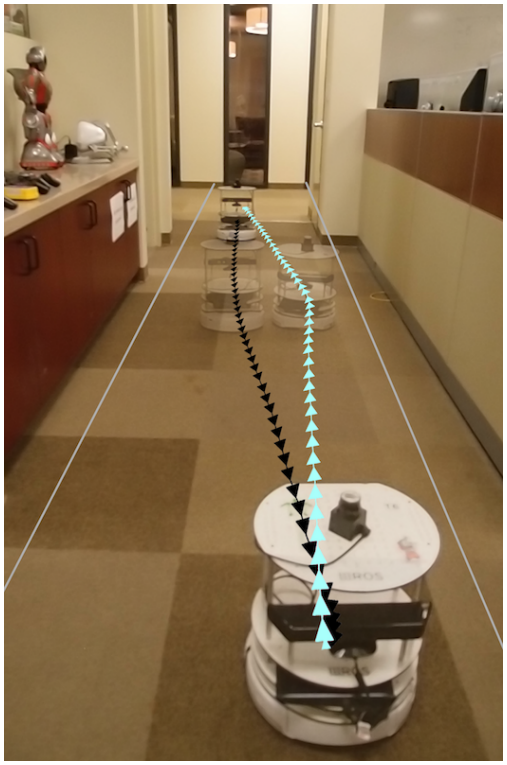


Figure 8: Real-world collision avoidance with two differential drive robots using CALU in a small hallway. To stay clear from the walls two extra ORCA lines where added to both robot.

7. REFERENCES

[1] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul Beardsley, and Roland Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In *Proceedings of the 10th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2010.

[2] Paolo Fiorini and Zvi Shiller. Motion planning in dynamic environments using velocity obstacles. *International Journal of Robotics Research*, 17:760–772, July 1998.

[3] Dieter Fox. Kld-sampling: Adaptive particle filters. In *Advances in Neural Information Processing Systems 14*. MIT Press, 2001.

[4] Dieter Fox. Adapting the sample size in particle filters through kld-sampling. *International Journal of Robotics Research*, 22, 2003.

[5] Boris Kluge, Dirk Bank, Erwin Prassler, and Matthias Strobel. Coordinating the motion of a human and a robot in a crowded, natural environment. In *Advances in Human-Robot Interaction*, volume 14 of *Springer Tracts in Advanced Robotics*, pages 231–234. Springer Berlin / Heidelberg, 2005.

[6] Morgan Quigley et al. ROS: An open-source Robot Operating System. In *Proceedings of the Open-Source Software workshop (ICRA)*, 2009.

[7] Jamie Snape, Jur van den Berg, Stephen J. Guy, and Dinesh Manocha. Independent navigation of multiple mobile robots with hybrid reciprocal velocity obstacles. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5917–5922, 2009.

[8] Jamie Snape, Jur van den Berg, Stephen J. Guy, and Dinesh Manocha. Smooth and collision-free navigation for multiple robots under differential-drive constraints. In *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2010.

[9] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents series)*. The MIT Press, 2005.

[10] Jur van den Berg, Stephen Guy, Ming Lin, and Dinesh Manocha. Reciprocal n-body collision avoidance. In *Robotics Research*, volume 70, pages 3–19, 2011.

[11] Jur van den Berg, Ming Lin, and Dinesh Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *ICRA 2008*, pages 1928 –1935, 2008.

[12] Richard Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2):189–208, 2008.

Decision-Theoretic Approach to Maximizing Observation of Multiple Targets in Multi-Camera Surveillance

Prabhu Natarajan, Trong Nghia Hoang, Kian Hsiang Low, and Mohan Kankanhalli
Department of Computer Science, National University of Singapore
Computing 1, 13 Computing Drive, Singapore 117417, Republic of Singapore
{prabhu, nghiaht, lowkh, mohan}@comp.nus.edu.sg

ABSTRACT

This paper presents a novel decision-theoretic approach to control and coordinate multiple active cameras for observing a number of moving targets in a surveillance system. This approach offers the advantages of being able to (a) account for the stochasticity of targets' motion via probabilistic modeling, and (b) address the trade-off between maximizing the expected number of observed targets and the resolution of the observed targets through stochastic optimization. One of the key issues faced by existing approaches in multi-camera surveillance is that of scalability with increasing number of targets. We show how its scalability can be improved by exploiting the problem structure: as proven analytically, our decision-theoretic approach incurs time that is linear in the number of targets to be observed during surveillance. As demonstrated empirically through simulations, our proposed approach can achieve high-quality surveillance of up to 50 targets in real time and its surveillance performance degrades gracefully with increasing number of targets. We also demonstrate our proposed approach with real AXIS 214 PTZ cameras in maximizing the number of Lego robots observed at high resolution over a surveyed rectangular area. The results are promising and clearly show the feasibility of our decision-theoretic approach in controlling and coordinating the active cameras in real surveillance system.

Categories and Subject Descriptors

I.4.8 [**Scene Analysis**]: Tracking; I.2.9 [**Robotics**]: Commercial robots and applications, Sensors

General Terms

Algorithms, Performance, Experimentation, Security

Keywords

Active camera networks, Smart camera networks, Multi-camera coordination and control, Surveillance and security

1. INTRODUCTION

The use of active cameras in surveillance is becoming increasingly popular due to the recent advances in smart camera technologies [4]. These active cameras are endowed with pan, tilt, and zoom capabilities, which can be exploited to provide high-quality surveillance. In order to achieve effective, real-time surveillance, an efficient collaborative mechanism is needed to control and coordinate these cameras' actions, which is the focus of our work in this paper.

Monitoring a set of targets moving in an environment is a challenging and difficult task because (a) the motion of these targets is often stochastic in nature, (b) it needs to address the non-trivial trade-off between maximizing the expected number of observed targets and the resolution of the observed targets, and (c) a camera coordination framework should be scalable with an increasing number of targets. To elaborate, (a) the uncertainty in the targets' motion makes it hard for the active cameras to know where to observe in order to keep these targets within their fields of view (fov) and they may consequently lose track of the observed targets, (b) increasing the resolution of observing some targets through panning, tilting, or zooming may result in the loss of other targets being tracked, and (c) when the number of targets increases, a camera coordination framework, if poorly designed, tends to incur exponentially increasing computational time, which degrades the performance of the entire system. These issues arise in many real-world surveillance applications such as target surveillance, observing a group of players in sports, industrial monitoring of protected sites, etc. Hence, we believe that, by addressing these practical issues, a more effective surveillance system can be realized and subsequently deployed in the real world. Note that our proposed surveillance task differs from the typical sensor coverage problem, the latter of which instead focuses on maximizing the spatial coverage of the cameras that are independent of targets' motion. In our work, we try to maximize the coverage of the observed targets in the environment.

This paper presents a novel principled decision-theoretic approach to control and coordinate the active cameras for the surveillance of multiple moving targets (Section 2). This approach is based on the Markov Decision Process (MDP) framework, which allows the surveillance task to be framed formally as a stochastic optimization problem (Sections 3 and 4). In particular, our MDP-based approach resolves the above-mentioned issues: (a) the motion of the targets can be modeled probabilistically (Section 4.2), and (b) to address the trade-off, the active cameras' actions are coordinated to maximize the expected number of observed targets while guaranteeing a pre-defined resolution of these observed targets (Section 4.4), and (c) the scalability can be improved by exploiting the problem structure: as proven analytically (Section 4.5), our MDP-based approach incurs time that is linear in the number of targets to be observed during surveillance. One key problem faced by existing multi-camera multi-target surveillance approaches is that of scalability with increasing number of targets (Section 2). As demonstrated empirically through simulations (Section 5), our MDP-based approach

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

Table 1: Comparison of related work based on (a) camera:target ratio, (b) primary criterion, and (c) uncertainty in targets' motion.

Surveillance/tracking strategy	$n \ll m$	$n \gg m$	$n = m$	Maximizing no. of observed targets	Minimizing uncertainty of targets' locations	Uncertainty in targets' motion
Banerjee et al. [3]			×		×	
Costello et al. [5]		×			×	
Krahnstoever et al. [9]		×			×	
Qureshi et al. [13]		×			×	
Soto et al. [15]		×			×	
Sommerlade et al. [14]		×			×	
Huang et al. [8]		×			×	
Alfy et al. [6]		×			×	
Proposed MDP-based approach	×			×		×

can achieve high-quality surveillance of up to 50 targets in real time and its surveillance performance degrades gracefully with an increasing number of targets. The real-world experiments (Section 5.3) show the practicality of our decision-theoretic approach to control and coordinate cameras in surveillance systems.

2. RELATED WORK

Our proposed work is compared and contrasted with existing approaches for active camera surveillance based on the following classification: (a) ratio of number n of cameras to number m of targets, (b) primary criterion - the main objective/goal of the surveillance system, and (c) uncertainty in targets' motion - whether the targets' motion uncertainty is considered in camera coordination and optimal decision making. This comparison is shown in Table 1. The camera:target ratio is further classified based on $n \ll m$, $n \gg m$, and $n = m$. The camera:target ratio plays an important role in the choice of primary criterion that is used in the existing works, as explained below. The primary criterion is classified into: (i) maximizing the number of observed targets with certain guaranteed resolution and (ii) minimizing the uncertainty of individual targets' locations. The targets' motion is stochastic in nature and hence needs to be predicted and subsequently exploited for coordinating the cameras in a typical surveillance system. The existing works are also classified based on whether they have accounted for the uncertainty in targets' motion in their optimization framework.

Table 1 shows that when the camera:target ratio is either $n = m$ [3] or $n \gg m$ [5, 6, 8, 9, 13, 14, 15], the primary criterion is to minimize the uncertainty of individual targets' locations. By observing individual targets with more cameras, the uncertainty of targets' locations is decreased. In contrast, when the camera:target ratio is $n \ll m$, the primary criterion is to maximize the number of observed targets in the environment. In either criterion, the targets' motion is inherently non-deterministic. But, none of the previous works have accounted for the motion uncertainty in their optimization framework. The works of [2, 12] aim to maximize the coverage of static targets in omni-directional active sensors. Since the targets are static, there is no notion of stochasticity of targets' motion. All the above-mentioned works use heuristic approaches to select the best actions for the active cameras. Such approaches are therefore tailored specifically to their own objectives and cannot be modified to achieve other objectives. In contrast, our approach is a general framework in which different surveillance goals can be modeled as formal objective functions.

To summarize, our proposed work is different from the existing works in the following ways: (a) we use a formal, principled Markov Decision Process (MDP) framework to select the optimal actions for active cameras to maximize the expected number of observed targets; (b) we account for the uncertainty in targets' motion by integrating a probabilistic motion model into our optimization framework; and (c) many previous works ([9, 13, 14, 16], etc.) face a serious scalability issue in terms of the number of targets to be observed. We shall show in later sections how the state space of the targets can be managed efficiently by exploiting the structure and properties that are inherent in the surveillance problem.

3. SYSTEM ARCHITECTURE

The proposed surveillance framework consists of a supervised surveillance environment and an MDP controller. The environment consists of targets, static cameras, and active cameras. The targets are the moving objects (e.g., people, vehicles, robots, etc.) in the surveillance environment whose motions are stochastic in nature. The static cameras are wide-view cameras that can only provide low-quality information of the surveillance environment. These cameras are assumed to be calibrated and can obtain the 3D location, direction, and velocity information of the targets. The active cameras are PTZ (pan/tilt/zoom) cameras that can get high-resolution images of the targets in the environment. The MDP controller models the interaction between the active cameras and the environment, and provides a platform to choose optimal actions for these cameras in order to achieve high-quality surveillance tasks.

Fig. 1 shows the top view of a representative surveillance environment where the full fov's of the active cameras are shown in dotted lines and the current active fov's are shaded. For simplicity, the static cameras are not shown. The active cameras are placed such that they can observe the complete environment by pan/tilt/zoom operations but cannot observe all locations of the environment simultaneously. This makes the problem more practical and challenging, thus emphasizing the need to control these active cameras. The static cameras determine the location, direction, and velocity of targets and pass these information to the MDP controller. Based on these information, the MDP controller computes the optimal actions of active cameras such that the expected utility of the surveillance system is maximized. The utility of the surveillance system corresponds to the high-level application goal that can be defined formally using a real-valued objective function, as described in Section 4.4.

Formally, the MDP controller is defined as a tuple $(\mathcal{S}, \mathcal{A}, R, T_f)$ consisting of a set $\mathcal{S}$ of discrete states of active cameras and targets, a set $\mathcal{A}$ of joint actions of active cameras, a reward function $R : \mathcal{S} \rightarrow \mathbb{R}$ representing the high-level surveillance goal, and a transition function $T_f : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ denoting the probability $P(S'|S, A)$ of switching from the current state $S \in \mathcal{S}$ to the next state $S' \in \mathcal{S}$ using the joint action $A \in \mathcal{A}$. In the MDP framework, the policy function $\pi : \mathcal{S} \rightarrow \mathcal{A}$ maps from each state to

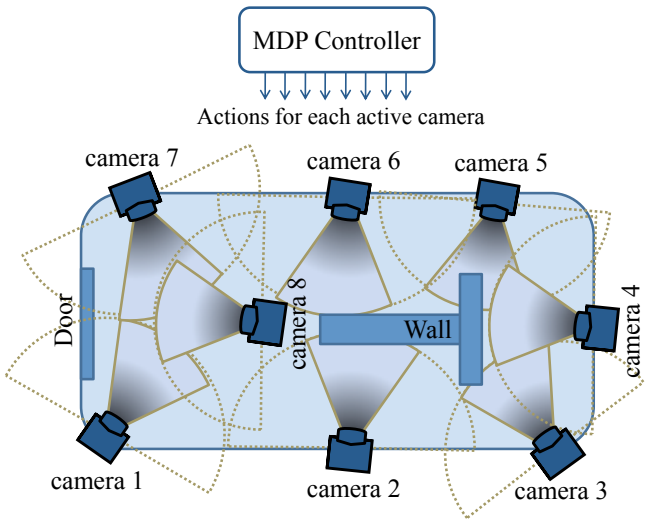


Figure 1: System architecture.

a joint action of the cameras. Solving the MDP involves choosing the policy that maximizes the expected reward for any given state. The optimal policy, denoted by π^*, maximizing the expected utility of the system in the next time step is given by

$$\pi^*(S) = \arg\max_{A \in \mathcal{A}} \sum_{S' \in \mathcal{S}} R(S')\, P(S'|S, A)\,.$$

The main challenge in the MDP is managing the state space $\mathcal{S}$ and action space $\mathcal{A}$. This is because the state space grows exponentially in the number of active cameras and targets. Hence, the policy computation time for our surveillance problem is exponential. In practice, the structure of the problem and environment can usually be exploited to reduce the number of states and the time required to compute the optimal policy. We will show in Section 4.5 how the state space can be managed for our surveillance problem, thus allowing the MDP to be solved more efficiently.

The following assumptions are made in our surveillance task:

- The targets are oblivious to the cameras, in particular, non-evasive (i.e., they do not try to escape from the cameras' fields of view) and their motion cannot be controlled nor influenced;
- The static cameras are calibrated accurately such that the 3D positioning errors of the targets are minimal. This can be achieved by placing the cameras at high altitude;
- The total number of targets in the environment can be obtained from static cameras and/or motion sensors at the entry and exit.

4. PROBLEM FORMULATION

Given a set of cameras and targets in a surveillance system, the MDP controller determines the optimal actions for these cameras such that the expected utility of the surveillance system is maximized. In this section, we describe how an MDP framework can be applied to a generic active camera surveillance in order to maximize the expected utility of the surveillance system. We enumerate each component of the MDP framework and show how these components can be formulated for a typical surveillance system. In this work, the objective/reward function of the MDP modeling the high-level surveillance goal measures the total number of targets observed by active cameras with a guaranteed resolution. Maximizing the number of observed targets with a guaranteed resolution is a mandatory task in surveillance because we need to obtain the high-resolution images of targets for biometric and forensic tasks

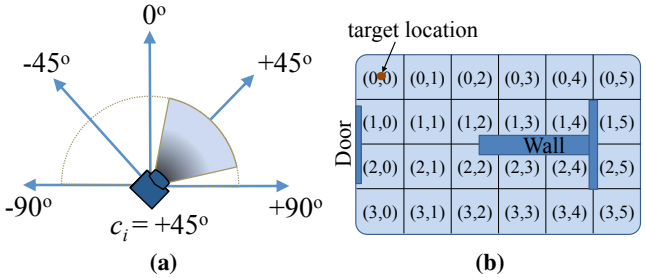


Figure 2: (a) Camera states and (b) target locations.

like target detection, recognition, etc. In this work, we present a decision-theoretic approach for maximizing the expected number of targets observed by the active cameras.

4.1 States and Actions

A state of the MDP comprises the states of active cameras and targets in the surveillance environment. The passive static cameras are first calibrated based on common ground plane coordinates and then used to obtain the targets' approximate 3D location, velocity, and the direction information. Let n be the number of active cameras and m be the number of targets in the environment such that $n \ll m$. In this manner, the surveillance problem becomes more challenging and interesting since there are more targets to be monitored by fewer active cameras.

Let the set of possible states of each active camera in the environment be denoted by $\mathcal{C}$ such that each state $c_i \in \mathcal{C}$ corresponds to a discretized pan/tilt/zoom position of camera i. For example, in Fig. 2a, the set of possible states of camera i based on discretized pan angles is given by $\mathcal{C} = \{+90\,^\circ, +45\,^\circ, 0\,^\circ, -45\,^\circ, -90\,^\circ\}$ and the current state c_i is $+45\,^\circ$.

Let the state space of a target be represented by a set of tuples of location, direction and velocity, and denoted by $\mathcal{T} = \mathcal{T}_l \times \mathcal{T}_d \times \mathcal{T}_v$ where $\mathcal{T}_l$ denotes a set of all possible locations of the target in the environment, $\mathcal{T}_d$ denotes a set of all possible discretized directions between all pairs of locations in $\mathcal{T}_l$, and $\mathcal{T}_v$ denotes a set of discretized velocities of the target. The surveillance environment is discretized into grid cells such that the centers of the grid cells represent the possible locations of a target, as shown in Fig. 2b. The approximate 3D location of the target observed by static cameras will be mapped to the center of the nearest grid cell. The direction and velocity of the target are determined based on its current and previous locations. The static cameras detect the targets in their fov's and report their locations, directions, and velocities to the MDP controller.

By calibrating the active cameras, the possible target locations in the environment that lie within the fov of each active camera in its various states can be pre-computed. For each state $c_i \in \mathcal{C}$ of active camera i, the subset of locations lying within its corresponding fov is denoted by $fov(c_i) \subset \mathcal{T}_l$. For example, Fig. 3 illustrates the fov (i.e., shaded polygon) of active camera 1 in its current state c_1; the subset of locations that are observed by camera 1 is given by $fov(c_1) = \{(0,1), (0,2), \ldots, (2,3), (2,4)\}$.

To observe targets with a guaranteed resolution, the zoom parameter of an active camera can be adjusted to focus its fov so that imageries of the targets detected within its fov satisfy a pre-defined resolution. This requires limiting the depth of its fov, as depicted by the horizontal line in Fig. 3. As a result, if a target is located within $fov(c_i)$ of any camera i, then it is observed with a guaranteed resolution. For example, the minimum resolution of the human face should be 24×24 pixels, which is the base resolution for face detection [18]. The resolution of the targets should be higher than

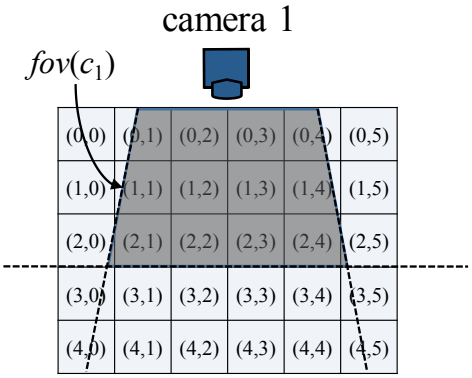


Figure 3: $fov(c_1)$ of camera 1.

24×24 pixels for other tasks like face recognition and expression analysis, vehicle number plate detection and identification, etc.

Let the vector $C = (c_1, c_2, \ldots, c_n)$ be the joint state of n active cameras in the environment and the vector $T = (t_1, t_2, \ldots, t_m)$ be the joint state of m targets in the environment where $t_k \in \mathcal{T}$ is the state of target k. A state $S \in \mathcal{S} = \mathcal{T}^m \times \mathcal{C}^n$ of the MDP is therefore of the form $S = (T, C)$.

The actions of an active camera are pan/tilt/zoom commands to move the camera to a specified state. Let a_i be an action of camera i corresponding to a pan/tilt/zoom command. We assume that the delay in moving the camera to a specified state is negligible as the state-of-the-art cameras are capable of panning at a speed of 360°/sec [1]. The joint action of all cameras at any given time is a vector $A = (a_1, a_2, \ldots, a_n) \in \mathcal{A}$. Since we assume that the targets' motion cannot be controlled, no action can be specified by the MDP controller to influence their motion in the surveillance environment.

4.2 Transition Function T_f

Recall that the transition function T_f of the MDP denotes the probability $P(S'|S, A)$ of moving from the current state S to the next state S' using the joint action A. In this subsection, we will show how this transition probability can be factored into transition probabilities of individual active cameras and targets using the conditional independence property, which is inherent in the state transition dynamics of the surveillance environment. As a result, the computation time of our optimal policy is significantly reduced (i.e., from exponential to linear in the number m of targets), hence alleviating the scalability issue (see Theorem 1).

Firstly, the transition probability $P(S'|S, A)$ can be factored into the transition probabilities of the active cameras and targets (i.e., respectively, $P(C'|C, A)$ and $P(T'|T)$) due to conditional independence (see first equality of (1)). Specifically, the transition probability $P(C'|C, A)$ of the active cameras is conditionally independent of the targets' states. Since the targets are assumed to be oblivious to the cameras, the transition probability $P(T'|T)$ (i.e., motion model) of the targets is conditionally independent of the active cameras' states and actions.

Next, the transition probability $P(C'|C, A)$ of the active cameras can also be factored into transition probabilities of individual active cameras due to conditional independence. The transition probability of an individual camera i is $P(c_i'|c_i, a_i)$ where $c_i, c_i' \in \mathcal{C}$ are, respectively, its current and next states, and a_i is its action. Since the transition probability of each active camera is conditionally independent of the other cameras given its current state and action, $P(C'|C, A)$ can be factored into $P(c_i'|c_i, a_i)$'s for $i = 1, \ldots, n$ (see second equality of (1)). Modern active cameras are equipped with advanced functionalities that enable them to move to the desired pan/tilt/zoom positions accurately [1]. Hence, it is practical to assume the transition of camera i to be deterministic and consequently represented by a deterministic function $c_i' = execute(c_i, a_i)$ since $P(c_i'|c_i, a_i)$ evaluates to either 0 or 1.

Similarly, the transition probability $P(T'|T)$ of the targets can be factored into transition probabilities (i.e., motion models) of individual targets by assuming conditional independence. The transition probability of target k is $P(t_k'|t_k)$ where $t_k, t_k' \in \mathcal{T}$ are, respectively, its current and next states. Since the transition probability of each target is conditionally independent of the other targets given its current state, $P(T'|T)$ can be factored into $P(t_k'|t_k)$'s for $k = 1, \ldots, m$ (see second equality of (1)).

As discussed above, the transition probability $P(S'|S, A)$ of the MDP can be factored into transition probabilities of individual active cameras and targets after repeatedly applying the conditional independence property:

$$\begin{aligned} & P(S'|S, A) \\ &= P(C'|C, A)\ P(T'|T) \\ &= \prod_{i=1}^{n} P(c_i'|c_i, a_i)\ \prod_{k=1}^{m} P(t_k'|t_k) \\ &= \begin{cases} \prod_{k=1}^{m} P(t_k'|t_k) & \text{if } P(c_i'|c_i, a_i) = 1 \text{ for } i = 1, \ldots, n, \\ 0 & \text{otherwise.} \end{cases} \end{aligned} \tag{1}$$

4.3 Transition Probability $P(t_k'|t_k)$ of a Target

To calculate the transition probability of a target, we first predict a target's movement in a surveillance environment using a general velocity-direction motion model. Specifically, this model comprises two Gaussian distributions for the velocity v and direction d of the target: $v \sim \mathcal{N}(\mu_v, \sigma_v)$ and $d \sim \mathcal{N}(\mu_d, \sigma_d)$ where the mean parameters μ_v and μ_d are obtained from the static cameras at every time step based on the previous location of the target, and the variance parameters σ_v and σ_d are learned from a dataset of targets' trajectories in the given supervised surveillance environment.

Then, in every time step t, we draw paired samples of velocity v and direction d of the target from the Gaussian distributions, compute its corresponding predicted location (x_t, y_t) in the environment using

$$\begin{aligned} x_t &= x_{t-1} + v \times \cos(d) \times \mathrm{d}t \\ y_t &= y_{t-1} + v \times \sin(d) \times \mathrm{d}t \end{aligned} \tag{2}$$

and determine the proportion of samples in each grid cell to produce the transition probability $P(t_k'|t_k)$ of the target. Fig. 4 shows the transition probability distribution of a target that is located at $(x_{t-1}, y_{t-1}) = (5, 5)$ with $\mu_v = 2$ cells per time step and $\mu_d = 45°$. The probability distribution of the neighboring locations that the target will move to in time step t is shown as black dots. Since the possible locations, directions, and velocities of the target are finite, we can pre-compute the transition probabilities of the target and store them off-line. This helps to reduce the on-line policy computation time, as discussed in Theorem 2.

4.4 Objective/Reward Function R

The advantage of using MDPs in surveillance systems is that any high-level surveillance goal can be defined formally using a real-valued objective/reward function. In this work, the goal of the surveillance system is to maximize the number of observed targets with a guaranteed resolution. Supposing the states of all targets are known, such a goal can be achieved by defining a reward function

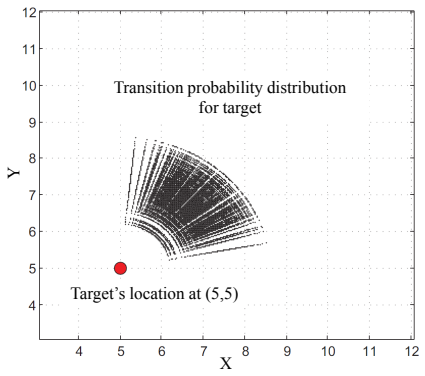


Figure 4: Transition probability distribution of a target.

that measures the total number of targets lying within the fov of any of the active cameras:

$$R(S) = R((T, C)) = \sum_{k=1}^{m} \widetilde{R}(t_k, C) \tag{3}$$

$$\widetilde{R}(t_k, C) = \begin{cases} 1 & \text{if target } k\text{'s location lies in } fov(C), \\ 0 & \text{otherwise;} \end{cases} \tag{4}$$

where $fov(C) = \bigcup_{i=1}^{n} fov(c_i)$ denotes a set of target locations in the environment, each of which lies within the fov of at least one active camera when the cameras are in state C. So, if the location of target k lies within $fov(C)$, then it is guaranteed to be observed at a predefined image resolution, as discussed in Section 4.1, and $\widetilde{R}(t_k, C) = 1$ results.

4.5 Policy Computation

The states of the targets in the next time step are uncertain due to stochasticity of their motion. Therefore, the optimal policy π^* has to instead maximize the *expected* total number of targets that lie within the fov of any of the active cameras in the next time step:

$$\pi^*(S) = \pi^*((T, C)) = \arg\max_{A \in \mathcal{A}} V(T, C, A) \tag{5}$$

$$V(T, C, A) = \sum_{T' \in \mathcal{T}^m} R((T', C'))\ P(T'|T) \tag{6}$$

where T' and C' are, respectively, the joint states of the targets and active cameras in the next time step. The next joint state C' of the cameras can be determined deterministically from their current joint state C and action A using the function $c_i' = execute(c_i, a_i)$ for $i = 1, \ldots, n$ (Section 4.2).

Computing the policy π^* (5) for a given state S incurs $\mathcal{O}(|\mathcal{A}||\mathcal{T}|^m)$ time, which is exponential in the number m of targets. Its time complexity can be significantly reduced by exploiting the inherent structure of our surveillance problem, in particular, the conditional independence property in the transition model of the MDP (Section 4.3). As a result, the value function V (6) can be reduced to

$$V(T, C, A) = \sum_{k=1}^{m} \widetilde{V}(t_k, C') \tag{7}$$

$$\widetilde{V}(t_k, C') = \sum_{t_k' \in \mathcal{T}} \widetilde{R}(t_k', C')\ P(t_k'|t_k)\ . \tag{8}$$

For a detailed derivation of (7), see Appendix A. Computing the policy π^* for a given state S consequently incurs linear time in the number m of targets, as shown in the result below:

THEOREM 1. *If (1) holds, then computing policy π^* (5) for a given state S incurs $\mathcal{O}(|\mathcal{A}||\mathcal{T}|m)$ time.*

To improve the real-time computation of policy π^*, the values of $\widetilde{V}(t_k, C')$ (8) for all $t_k \in \mathcal{T}$ and $C' \in \mathcal{C}^n$ can be pre-computed and stored off-line. To do this, the values of $P(t_k'|t_k)$ for all $t_k, t_k' \in \mathcal{T}$ have to be pre-computed first, which incurs $\mathcal{O}(|\mathcal{T}|^2)$ time. The values of $\widetilde{R}(t_k', C')$ for all $t_k' \in \mathcal{T}$ and $C' \in \mathcal{C}^n$ also have to be pre-computed, which incurs $\mathcal{O}(|\mathcal{T}||\mathcal{C}|^n)$ time. Consequently, the values of $\widetilde{V}(t_k, C')$ (8) for all $t_k \in \mathcal{T}$ and $C' \in \mathcal{C}^n$ can be pre-computed in $\mathcal{O}(|\mathcal{T}|^2|\mathcal{C}|^n)$ time. Hence, the total off-line computation time is $\mathcal{O}(|\mathcal{T}|^2|\mathcal{C}|^n)$. The on-line computation time to derive policy π^* can then be reduced to $\mathcal{O}(|\mathcal{A}|m)$, which includes the time taken to look up the values of $\widetilde{V}(t_k, C')$ for m targets (7) and over $|\mathcal{A}|$ possible joint actions (5). The result below summarizes the computation time incurred by the on-line and off-line processing steps:

THEOREM 2. *If (1) holds, then computing policy π^* (5) for a given state S incurs off-line computation time of $\mathcal{O}(|\mathcal{T}|^2|\mathcal{C}|^n)$ and on-line computation time of $\mathcal{O}(|\mathcal{A}|m)$.*

5. EXPERIMENTS AND DISCUSSIONS

In this section, we present empirical evaluation of our MDP-based approach for maximizing the number of targets observed by active cameras. Our proposed approach is simulated in Player/Stage simulator [7] to perform extensive experimentations and implemented using real AXIS 214 PTZ cameras to demonstrate its feasibility in real surveillance system. Before describing them, it is important to point out that there is no standard benchmark surveillance environments and datasets for active camera networks to compare our proposed approach with the other systems in the literature (e.g., [9, 13, 14]). While the primary criterion of these systems is to minimize the uncertainty of targets' locations, our objective function is to maximize the number of targets observed in high-resolution images (see Table 1). These existing systems use heuristic approaches that can optimize only their respective objective function and cannot be used for other objective functions. These systems also suffer from scalability issue when the number of targets is increased. Furthermore, the optimization frameworks of these existing systems determine the cameras' actions (or schedule the cameras) for the current time step based on the *current* locations of the targets. In contrast, our approach determines the cameras' actions for the current time step based on the *expected* locations of the targets in the next time step (see (7) and (8)). This makes our approach perform better than the existing methods in maximizing the number of observed targets. Our MDP-based approach is empirically compared with the following existing heuristic methods:

- *Krahnstoever's (Krahns) Approach*: The work of [9] provided an optimization method to capture high-resolution images of a single target. It schedules the tasks for active cameras based on the location of the targets in the current time step and assumes that the targets will not move out of the fov within the short duration;
- *Systematic (Sys) Approach*: The active cameras pan automatically in a round robin fashion such that every camera pans to each of its states for a finite duration;
- *Static (Stat) Approach*: The active cameras are fixed at specific states such that they can cover maximum area to get high-resolution imageries of the targets.

Our approach and the above heuristic methods are evaluated using the following performance metric:

$$PercentObs = \frac{100}{\tau M_{tot}} \sum_{i=1}^{\tau} M_{obs}^i$$

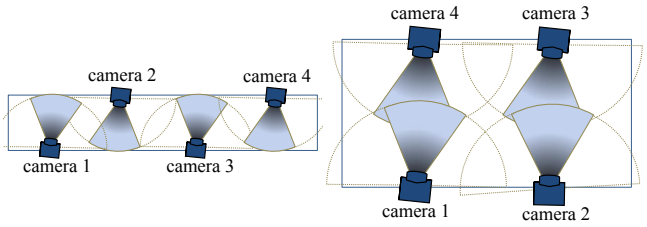


Figure 5: Setups of corridor and hall environments.

where τ (i.e., set to 100 in simulations) is the total number of time steps taken in our experiments, M^i_{obs} is the total number of targets observed by the active cameras at a given time step i, and M_{tot} is the total number of targets present in the environment. That is, the $PercentObs$ metric averages the percentage of targets being observed by the active cameras over the entire duration of τ time steps. We will first discuss the environmental setup for the simulated experiments and analyze the experimental results. Then, we will show the results of the real camera experiments. Interested readers can view our demo video[1].

5.1 Simulated Experiments: Setup

In Player/Stage simulator, we have designed an active camera model with functionalities to simulate real active cameras by configuring the number of states across pan angles, as discussed in Section 4.1. The targets' motion are generated in Player/Stage simulator based on velocity-direction motion model (see (2)), which resembles real human motion in surveillance environment. The locations of the targets are determined by a static camera, which is the simulator itself. We have conducted our experiments for two environmental setups (Fig. 5): corridor and hall. The sizes of the corridor and hall environments are, respectively, 40×5 grid cells and 20×10 grid cells such that $|\mathcal{T}_l| = 200$. The size of a grid cell in the simulator is approximately mapped to 1 m^2 in real world. We have used up to $n = 4$ active cameras with $|\mathcal{C}| = 3, 5$, and tested up to $m = 50$ targets. We have also conducted experiments for the camera resolutions $|fov(c_i)| \approx 25, 16$ by reducing the size of the camera's fov polygon in the simulator. The set $fov(c_i)$ of target locations that are observed by each active camera is determined by calibrating the active cameras in each of its state.

5.2 Simulated Experiments: Results

Figs. 6 and 7 show the performance of our MDP-based approach for corridor and hall setups with $n = 4$, $|\mathcal{T}_l| = 200$, target's velocity $v = 3$ cells per time step, and with varying m, $|fov(c_i)|$, $|\mathcal{C}|$, and sizes of clusters of targets that follow the Poisson distribution ($\lambda = 3$). The rest of this subsection describes the observations from our experiments.

Our MDP-based approach performs better for any of the target's velocity $v = 1, 2, 3$ cells per time step. This is because the cameras are controlled based on the predicted locations of a target by matching its corresponding transition probabilities with respect to its observed state. It can be observed from the experiments that the performance of MDP is much better that the other approaches when (a) the velocity of the targets is higher, (b) the targets move in clusters, and (c) when the resolution of the cameras is increased (i.e., $|fov(c_i)|$ is decreased). This is because when the velocity of the targets is high (i.e., $v = 2.5, 3$ cells per time step), all the targets will almost certainly move out of the fov's of the cameras in $Krahns$ approach as the cameras are controlled based on the current location of the targets, hence producing worse performance (see Figs. 6a and 7a). When the targets move in clusters, then the $Krahns$ approach suffers even more performance degradation because it has high tendency to lose clusters of targets. On the other hand, since the MDP has the correct transition model, it gives superior performance even when the targets move in high velocity. By increasing the resolution of the active cameras (i.e., by reducing $|fov(c_i)| \approx 25$ to 16), it can be observed that the MDP performs much better when compared to the $Krahns$ approach (Figs. 6c, 6d, 7c, and 7d). This is because when the targets are moving at a velocity of $v = 3$ cells per time step and are observed at higher resolution (i.e., $|fov(c_i)|$ is smaller), the chance of losing the targets is high when the cameras are controlled based on current observed locations of the targets. Since MDP has transition model that predicts the next locations of the targets, it outperforms the other approaches when the targets are clustered and the resolution of the cameras is high.

The Sys and $Stat$ approaches perform worse in almost all cases except when $|fov(c_i)| \approx 16$ (Figs. 6c, 6d, 7c, and 7d) where the Sys approach performs better than $Krahns$ approach. This is due to the fact that the cameras are controlled independently of the targets' information in both Sys and $Stat$ approaches. This shows that the targets' information (e.g., location, direction, etc) play a vital role in achieving high-quality surveillance. But, when $|fov(c_i)| \approx 16$, the Sys approach performs slightly better than $Krahns$ because the chance of targets moving out of the fov is higher in $Krahns$ approach if the velocity of the targets is $v = 3$ cells per time step and the fov is reduced to $|fov(c_i)| \approx 16$. In all cases, the MDP outperforms the Sys and $Stat$ approaches.

When the number of states of each camera is increased from $|\mathcal{C}| = 3$ (Figs. 6c and 7c) to $|\mathcal{C}| = 5$ (Figs. 6d and 7d), the performance improves because more targets can be observed due to the additional camera states. The MDP-based approach performs better than the other approaches even when the transition model is inaccurate. This is tested by keeping the velocity of the targets moving at $v = 3$ cells per time step and matching the transition probabilities computed with velocities $v = 2, 2.5, 3$ cells per time step (Figs. 6c, 6d, 7c, and 7d). The performance of MDP computed with inaccurate transition probabilities is still much better than the other approaches. This is because the reward function is optimized with respect to the expected locations of the targets.

When the number of cameras is increased from $n = 2, 3$ to 4, the increase in performance of MDP is much better than the other approaches for $m < 10$ targets and comparable to (if not better than) other approaches for $m > 10$. This is because the prediction capability of our approach outperforms the other approaches with every addition of a new camera. The graph with increasing number of cameras is not shown here due to space limitation.

From these observations, we find that our MDP-based approach performs better than the other tested approaches in all the cases due to its prediction capability. Specifically, it outperforms $Krahns$ approach when the velocity of the targets and the resolution of the cameras are high.

5.3 Real Experiments

We have conducted real experiments with $n = 3$ AXIS 214 PTZ cameras to monitor up to $m = 6$ Lego robots (targets) in an environment with the size of $|\mathcal{T}_l| = 11 \times 9$ grid cells. The size of each grid cell is 0.5 m^2. Each camera has $|\mathcal{C}| = 3$ states. The states of the cameras are determined such that all the cells of the environment can be observed at high resolution by at least one camera. Given any joint state C of the cameras, only a subset of cells in the environment can be observed by these cameras, i.e., $fov(C) \subset \mathcal{T}_l$.

[1]http://www.comp.nus.edu.sg/~lowkh/camera.html

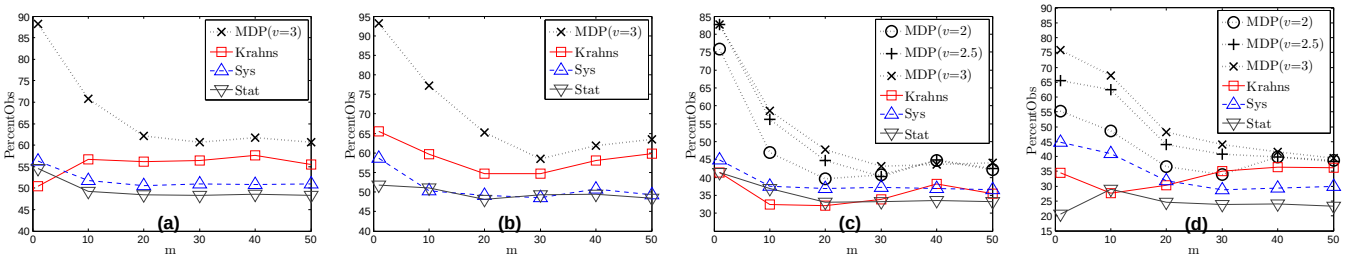


Figure 6: Graphs of $PercentObs$ vs. number m of targets for *corridor setup*: (a) non-clustered targets with $|fov(c_i)| \approx 25$ cells, $|\mathcal{C}| = 3$ and clustered targets with (b) $|fov(c_i)| \approx 25$ cells, $|\mathcal{C}| = 3$, (c) $|fov(c_i)| \approx 16$, $|\mathcal{C}| = 3$, (d) $|fov(c_i)| \approx 16$, $|\mathcal{C}| = 5$.

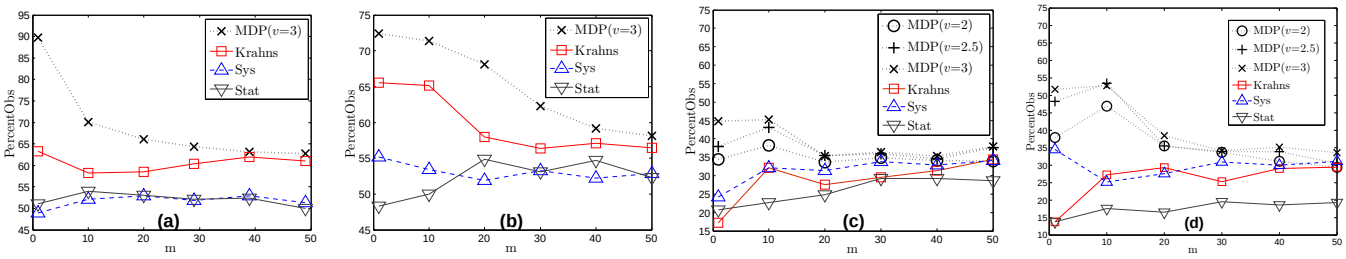


Figure 7: Graphs of $PercentObs$ vs. number m of targets for *hall setup*: (a) non-clustered targets with $|fov(c_i)| \approx 25$ cells, $|\mathcal{C}| = 3$ and clustered targets with (b) $|fov(c_i)| \approx 25$ cells, $|\mathcal{C}| = 3$, (c) $|fov(c_i)| \approx 16$, $|\mathcal{C}| = 3$, (d) $|fov(c_i)| \approx 16$, $|\mathcal{C}| = 5$.

This makes the problem challenging for the active cameras to maximize the number of observed robots. We have a static camera that can track these robots based on OpenCV Camshift tracker. The static camera is calibrated using [17] to obtain the approximate locations of the robots at every time step. The direction and velocity of the robots are determined based on their previous and current locations. The $fov(C)$ is determined by calibrating the active cameras in each of its state and determining the grid cells of the environment in which the robots can be observed at a high resolution. We guarantee the resolution of the robots that are observed by the active cameras to be approximately more than 40×40 pixels. We pre-computed the transition probabilities of an individual target for all possible locations, directions, and velocities $v = 1, 2$ cells per time step. The robots are moved based on the velocity-direction motion model and are programmed to turn back or stop when they hit the wall or cross other robots. Each robot is initialized with a Camshift tracker in the static camera and is tracked to get its approximate 3D location, direction, and velocity.

We have tested our implementation up to $m = 6$ robots but we keep one of the robots static. It can be observed that cameras 2 and 3 coordinate to observe the brown static robot (Fig. 8). Camera 2 pans to another state (see bottom two rows of Fig. 8) only when camera 3 takes over the observation of the static target (see top two rows of Fig. 8). This static target can be replaced by a portion of the surveillance environment like the entrance/exit or reception where we need to pay more attention. Table 2 shows the $PercentObs$ performance for the real experiments over $\tau = 50$ time steps.

Our proposed approach has some limitations: (a) only when the observations from static cameras are near-deterministic (i.e., with the help of overhead static cameras), our proposed approach is expected to perform well; (b) MDP observes its targets less well when their motion is more uncertain. In our future work, we will include an observation model to handle location uncertainty due to static cameras, which results in a Partially Observable Markov Decision Process framework. We will also look into deploying active cameras with a team of mobile robots [10, 11] for tracking and surveillance of mobile targets.

Table 2: Performance for real experiments.

m	1	2	3	4	5	6
$PercentObs$	99.2	97	95.3	93.5	88	85.1

6. CONCLUSION

This paper describes a novel decision-theoretic approach to control and coordinate multiple active cameras for observing a number of moving targets in a surveillance system. Specifically, it utilizes the Markov Decision Process framework, which accounts for the stochasticity of targets' motion via a probabilistic motion model and addresses the trade-off by maximizing the expected number of observed targets with a guaranteed resolution via stochastic optimization. The conditional independence property, which is inherent in our surveillance problem, is exploited in the transition model of the MDP to reduce the exponential policy computation time to linear time. As shown in simulations, our approach can scale up to 50 targets in real time. We have also implemented our proposed decision-theoretic approach using real AXIS 214 PTZ cameras to demonstrate its feasibility in real surveillance system.

7. REFERENCES

[1] AXIS 232D+ Network Dome Camera datasheet (http://www.axis.com).

[2] J. Ai and A. A. Abouzeid. Coverage by directional sensors in randomly deployed wireless sensor networks. *J. Comb. Optim.*, 11(1):21–41, 2006.

[3] S. Banerjee, A. Chowdhury, and S. Ghosh. Video surveillance with PTZ cameras: The problem of maximizing effective monitoring time. In K. Kant, S. V. Pemmaraju, and K. M. Sivalingam, editors, *ICDCN 2010*, volume 5935 of *LNCS*, pages 341–352. Springer-Verlag, 2010.

[4] A. N. Belbachir, editor. *Smart Cameras*. Springer, 2010.

[5] C. Costello and I.-J. Wang. Surveillance camera coordination through distributed scheduling. In *Proc. CDC*, 2005.

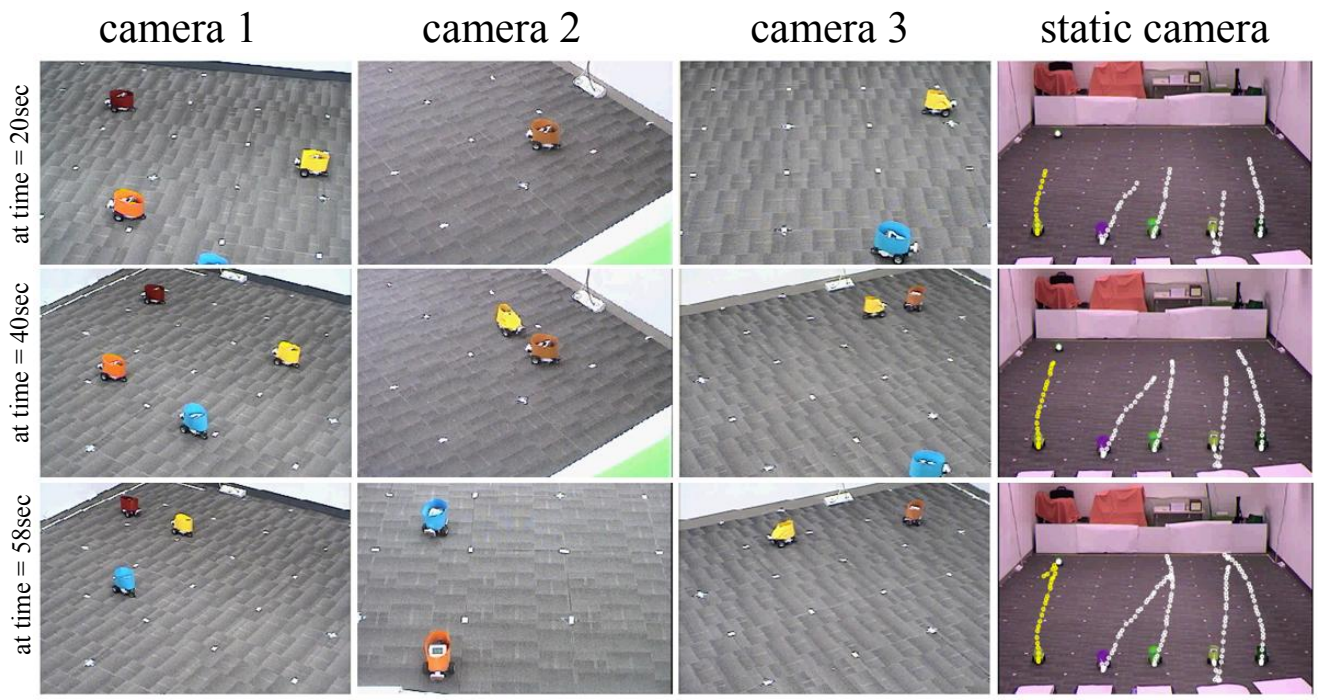


Figure 8: Results of real experiments: columns 1 to 3 show the high-resolution images of Lego robots captured by cameras 1, 2, and 3 while column 4 shows the targets' trajectories tracked by the static camera.

[6] H. El-Alfy, D. Jacobs, and L. Davis. Assigning cameras to subjects in video surveillance systems. In *Proc. ICRA*, 2009.

[7] B. P. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proc. ICAR*, pages 317–323, 2003.

[8] C.-M. Huang and L.-C. Fu. Multitarget visual tracking based effective surveillance with cooperation of multiple active cameras. *IEEE Trans. Syst., Man, Cybern. B*, 41(1):234 –247, 2011.

[9] N. Krahnstoever, T. Yu, S.-N. Lim, K. Patwardhan, and P. Tu. Collaborative Real-Time Control of Active Cameras in Large Scale Surveillance Systems. In *Proc. M2SFA2*, 2008.

[10] K. H. Low, W. K. Leow, and M. H. Ang, Jr. Task allocation via self-organizing swarm coalitions in distributed mobile sensor network. In *Proc. AAAI*, pages 28–33, 2004.

[11] K. H. Low, W. K. Leow, and M. H. Ang, Jr. Autonomic mobile sensor network with self-coordinated task allocation and execution. *IEEE Trans. Syst., Man, Cybern. C*, 36(3):315–327, 2006.

[12] V. P. Munishwar and N. B. Abu-Ghazaleh. Scalable target coverage in smart camera networks. In *Proc. ICDSC*, 2010.

[13] F. Qureshi and D. Terzopoulos. Planning ahead for PTZ camera assignment and handoff. In *Proc. ICDSC*, 2009.

[14] E. Sommerlade and I. Reid. Probabilistic surveillance with multiple active cameras. In *Proc. ICRA*, 2010.

[15] C. Soto, B. Song, and A. K. Roy-Chowdhury. Distributed multi-target tracking in a self-configuring camera network. In *Proc. CVPR*, pages 1486–1493, 2009.

[16] M. T. J. Spaan and P. U. Lima. A decision-theoretic approach to dynamic sensor selection in camera networks. In *Proc. ICAPS*, pages 279–304, 2009.

[17] R. Y. Tsai. An efficient and accurate camera calibration technique for 3D machine vision. In *Proc. CVPR*, 1986.

[18] P. Viola and M. J. Jones. Robust real-time face detection. *IJCV*, 57(2):137–154, 2004.

APPENDIX

A. PROOFS

Derivation of Equation 7

The value function V (6) is given by

$$
\begin{aligned}
& V(T, C, A) \\
&= \sum_{T' \in \mathcal{T}^m} R((T', C'))\ P(T'|T) \\
&= \sum_{t'_1 \in \mathcal{T}, \ldots, t'_m \in \mathcal{T}} \sum_{k=1}^{m} R(t'_k, C') \prod_{i=1}^{m} P(t'_i|t_i) \\
&= \sum_{k=1}^{m} \sum_{t'_k \in \mathcal{T}} R(t'_k, C')\ P(t'_k|t_k) \sum_{T'_{-k} \in \mathcal{T}^{m-1}} \prod_{i \neq k} P(t'_i|t_i) \\
&= \sum_{k=1}^{m} \sum_{t'_k \in \mathcal{T}} R(t'_k, C')\ P(t'_k|t_k) \\
&= \sum_{k=1}^{m} \widetilde{V}(t_k, C')
\end{aligned}
$$

where $T'_{-k} = (t'_1, \ldots, t'_{k-1}, t'_{k+1}, \ldots, t'_m)$. The second equality is obtained using (1) and (3). The fourth equality follows from

$$
\sum_{T'_{-k} \in \mathcal{T}^{m-1}} \prod_{i \neq k} P(t'_i|t_i) = \sum_{T'_{-k} \in \mathcal{T}^{m-1}} P(T'_{-k}|T_{-k}) = 1\ .
$$

Segregation in Swarms of e-puck Robots Based On the Brazil Nut Effect

Jianing Chen, Melvin Gauci, Michael J. Price and Roderich Groß
Natural Robotics Lab
Department of Automatic Control and Systems Engineering
The University of Sheffield, UK
{j.n.chen, m.gauci, r.gross}@sheffield.ac.uk, michaelprice@theiet.org

ABSTRACT

When a mixture of particles with different attributes undergoes vibration, a segregation pattern is often observed. For example, in muesli cereal packs, the largest particles—the Brazil nuts—tend to end up at the top. For this reason, the phenomenon is known as the Brazil nut effect. In previous research, an algorithm inspired by this effect was designed to produce segregation patterns in swarms of simulated agents that move on a horizontal plane.

In this paper, we adapt this algorithm for implementation on robots with directional vision. We use the e-puck robot as a platform to test our implementation. In a swarm of e-pucks, different robots mimic disks of different sizes (larger than their physical dimensions). The motion of every robot is governed by a combination of three components: (i) attraction towards a point, which emulates the effect of a gravitational pull, (ii) random motion, which emulates the effect of vibration, and (iii) repulsion from nearby robots, which emulates the effect of collisions between disks. The algorithm does not require robots to discriminate between other robots; yet, it is capable of forming annular structures where the robots in each annulus represent disks of identical size.

We report on a set of experiments performed with a group of 20 physical e-pucks. The results obtained in 100 trials of 20 minutes each show that the percentage of incorrectly-ordered pairs of disks from different groups decreases as the size ratio of disks in different groups is increased. In our experiments, this percentage was, on average, below 0.5% for size ratios from 3.0 to 5.0. Moreover, for these size ratios, all segregation errors observed were due to mechanical failures that caused robots to stop moving.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms, Experimentation, Reliability

Keywords

Brazil nut effect, Collective intelligence, Emergent behavior,

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

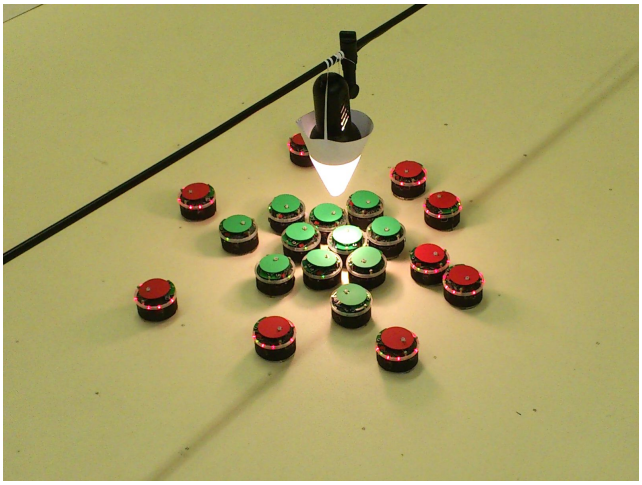


Figure 1: A segregation pattern in a swarm of 20 e-puck robots. The robots have organized into a center-periphery pattern around a light bulb. Robots with green and red top markers emulate disks of radius 8 cm and 16 cm, respectively. Each robot's motion is governed by a combination of three components: (i) attraction towards the light bulb, (ii) random motion, and (iii) repulsion from nearby robots.

Multi-robot systems, Segregation, Self-organization

1. INTRODUCTION

Segregation is a process whereby objects or individuals separate into distinct groups. It can be observed on various scales, ranging from the molecular to the macroscopic scale.

In this paper, we consider forms of segregation that are driven by self-organized processes [6]. We focus on the problem of making a swarm of physical robots self-organize into an annular structure. The robots are all identical in hardware; yet, by executing different behaviors, they are able to form a center-periphery pattern, as shown in Figure 1. We restrict our study to segregation between two groups of robots, mainly because of the limited number of physical robots available. However, the algorithm we use can, in principle, form annular structures with an arbitrary number of groups (and thus layers).

The formation of annular structures, and of center-periphery patterns in particular, might be useful in a range of

applications. Examples include reconfigurable nested membrane structures in biomedical applications and dynamically constructed defense structures in military applications.

A number of studies have looked at spatial segregation using simulated robotic agents. For example, Şahin *et al.* [8] implemented a control law based on a probabilistic framework. Kumar *et al.* [13] implemented a control law based on artificial potential functions. In these studies, segregation is the result of "individual choices that discriminate" [18]. In contrast, our robots are anonymous, and thus unable to discriminate between each other.

Other studies have looked at spatial segregation in the context of macroscopic self-assembly [11]. Bowden *et al.* [5] observed center-periphery structures when millimeter scale objects of two different heights interacted with each other by lateral capillary forces. Ngouabeu *et al.* [16] observed segregation phenomena in a system of vibrating and non-vibrating mechatronic modules that float on the surface of water.

Segregation phenomena observed in ant colonies [10] have inspired the implementation of control laws for robots that organize two distinct groups of items into center-periphery patterns [20, 14] (see also [1]). Unlike these works, our robots segregate themselves and are unable to discriminate between robots (or items) of different groups.

The Brazil nut effect [17] refers to the segregation that occurs when shaking a mixture of granular material of different sizes. Barker and Grimson [3] explain it as follows: "During the periods when shaking loosens the packing, individual small particles can move into voids beneath large particles and so prevent them from returning to their previous positions. It is far less probable that several small particles will move together so as to create a void that can be occupied by a single large particle. The net effect is that the smaller particles occupy the lower positions during the active part of the shaking process and then become trapped there when the grains fix into a new arrangement".

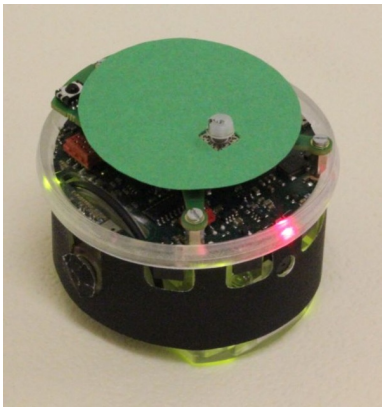
In previous research, a segregation algorithm based on the Brazil nut effect was developed and tested in computer simulation [12]. This algorithm assumed that every robot can instantly measure the relative position of all the robots in its vicinity. Here, we show how this algorithm can be modified to allow for an implementation using directional vision. This implies that (i) robots have to revolve in order to obtain an omni-directional picture and (ii) the algorithm has to cope with misperceptions, for example, due to visual occlusion (see Figure 5). We report on a series of experiments using the modified algorithm that show near error-free segregation in a swarm of 20 physical robots.

2. METHODS

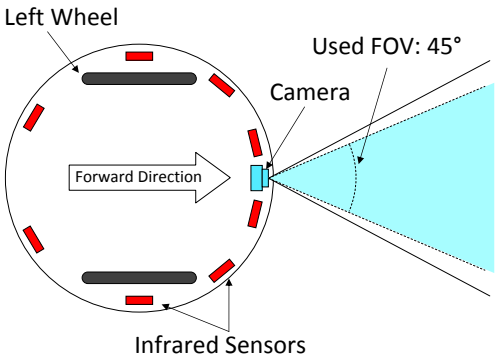
2.1 e-puck Robot

We use a mobile robot called *e-puck* (see Figure 2), which was developed for educational and research purposes [15]. It has a circular body of approximately 7.5 cm diameter, and weighs approximately 150 g. The e-puck is a differential-wheeled robot, having an inter-wheel distance of 5.1 cm.

In order to facilitate visual detection of robots by each other, we fitted every e-puck with a black paper skirt. Moreover, in order to allow for tracking of different groups of robots using an overhead camera system, we fitted the e-pucks with color-coded top markers. Figure 2(a) shows an e-puck fitted with a skirt and a green marker.



(a)



(b)

Figure 2: The e-puck robot. (a) An e-puck fitted with a black skirt and a green top marker. (b) Top-view schematic of an e-puck, indicating the locations of its wheels, camera [including the field of view (FOV)] and infrared sensors.

The e-puck has an RGB color camera located at its front. The camera has a resolution of 640 × 480 pixels (width × height), but the image taken was subsampled to 40 × 15 pixels. The e-puck also has eight infrared (IR) sensors, which are distributed around its body. Here, they are used in a passive mode, in order to detect the angular position of a light bulb within the arena. Figure 2(b) shows a top view of an e-puck, indicating the locations of the wheels, camera and IR sensors.

The robot has an IR receiver which allows IR signals to be sent to it, for example using an IR remote control. Here, we make use of this receiver in order to issue a starting signal to all of the robots at the beginning of every trial.

2.2 Controller

The controller used here is based on the one presented in [12]. Some modifications had to be made in order to port the algorithm onto physical e-puck robots. In the following, we describe the algorithm used here and highlight the modifications made.

The robots emulate a mixture of differently-sized disks subjected to vibration on a 2-dimensional plane. In partic-

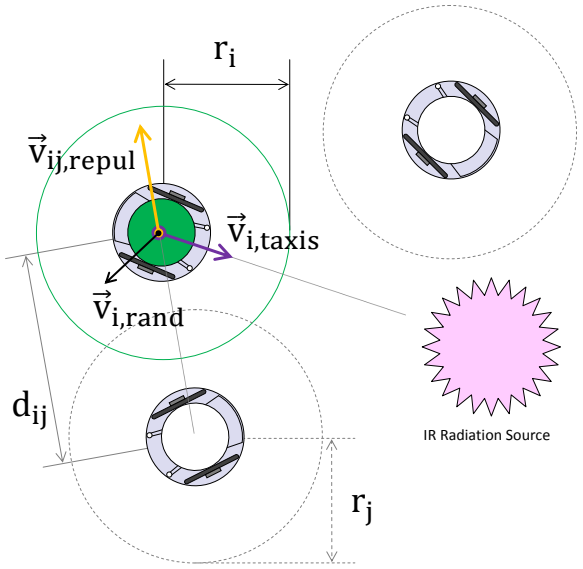


Figure 3: The three behavioral components of robot i. Vector $\vec{v}_{i,\text{taxis}}$ points towards the estimated location of the infrared radiation source. Vector $\vec{v}_{i,\text{rand}}$ points in a random direction. Vector $\vec{v}_{ij,\text{repul}}$ is due to the repulsion effect on robot i by robot j. Robot i is repelled by robot j if it perceives the virtual body of robot j as intersecting with its own virtual body. As robot i has no means of measuring the virtual radius of robot j (r_j), it assumes that $r_j = r_i$.

ular, robot i emulates a disk of radius r_i, whose motion is governed by a combination of three components (see Figure 3):

1. $\vec{v}_{i,\text{taxis}}$: attraction towards a point common to all the disks, which emulates the effect of a gravitational pull,
2. $\vec{v}_{i,\text{rand}}$: random motion, which emulates the effect of vibration and
3. $\vec{v}_{i,\text{repul}}$: repulsion from nearby disks, which emulates the effect of collisions.

Hereafter, the disk a robot represents is also referred to as the *virtual body* of the robot. The radius of the disk is also referred to as the *virtual radius* of the robot.

The behavior is implemented using the motor schema paradigm [2]. In every control cycle, robot i calculates the aforementioned three vectors. These are then combined as follows:

$$\vec{v}_i = \vec{v}_{i,\text{taxis}} + c_{\text{rand}}\vec{v}_{i,\text{rand}} + f(\vec{v}_{i,\text{repul}}). \quad (1)$$

Vector $\vec{v}_{i,\text{taxis}}$ is always a unit vector. Vector $\vec{v}_{i,\text{rand}}$ is also a unit vector but a parameter c_{rand} is used to weight its magnitude. Vector $\vec{v}_{i,\text{repul}}$ can have a large magnitude because it is computed as a sum of possibly many vectors (for details, see Section 2.2.3); therefore, its magnitude is capped by function $f(\cdot)$. Here, we use $c_{\text{rand}} = 0.6$ and a maximum allowed magnitude of 6.4 units for $\vec{v}_{i,\text{repul}}$. These settings follow suggestions from simulation results[1] [12].

[1]The algorithm in [12] uses an additional parameter to weight $\vec{v}_{i,\text{repul}}$. This is not used here because the repulsion mechanism has been modified. The weightings used here are identical to [12] when one considers the maximum allowed magnitude of $\vec{v}_{i,\text{repul}}$.

After constructing motion vector $\vec{v}_i$, robot i first turns to point in its direction, and then moves forward for a fixed duration. The speed at which it moves forward is proportional to the magnitude of the vector, so that the maximum magnitude possible (i.e., $1+0.6+6.4 = 8$ units) corresponds to the maximum speed of the robot (12.8 cm/s).

The length of the control cycle used here is 5 s, which is substantially longer than that used in simulation (0.1 s). The main reason for this is that the e-puck robots are equipped with directional cameras, whereas the simulated robots had omni-directional perception [12]. In each cycle, the robot spends around 2.4 s in revolving to obtain an omni-directional image, 1.3 s in turning to point in the direction of $\vec{v}_i$, and 1.3 s in moving forward.

In the following, we detail how vectors $\vec{v}_{i,\text{taxis}}$, $\vec{v}_{i,\text{rand}}$ and $\vec{v}_{i,\text{repul}}$ are computed.

2.2.1 Attraction to Center of Gravity

The algorithm requires a point of attraction in the environment to emulate the effect of a gravitational pull. Each robot is required to estimate the angular position of this point (the distance to it is not needed).

In our experimental setup, we use an infrared radiation source—a light bulb—as the point of attraction. In order to estimate its angular position, each robot makes use of its eight infrared sensors. In every control cycle, the three sensors giving the highest readings are selected. Each reading is then represented as a vector pointing from the center of the robot to the physical location of the sensor, with a magnitude proportional to the sensor's reading. The three vectors are summed, and the resulting vector is normalized to have a unit magnitude, giving $\vec{v}_{i,\text{taxis}}$.

2.2.2 Random Motion

The random motion vector $\vec{v}_{i,\text{rand}}$ is taken to be a unit vector pointing in a random direction in the interval $(0, 2\pi]$. This direction is taken with respect to the robot's orientation at the beginning of the control cycle.

2.2.3 Repulsion

In principle, each robot should be repelled by every other robot whose virtual body overlaps with its own virtual body. This would require the robots to know the virtual radii of nearby robots. However, as shown in [12], segregation can still be effectively achieved if every robot assumes for all other robots a constant virtual radius, which is a parameter that needs to be fixed a priori. Here, we propose and use an alternative, parameter-free heuristic: robot i assumes that the virtual radius of all other robots is equal to its own, that is, r_i.

In our implementation each robot uses its camera to estimate the angular position of and distance to nearby robots. In every control cycle, a robot turns through one revolution in eight steps of 45° each. In each step, its camera takes a picture. From the center of this picture, a horizontal line of 32 pixels is extracted (corresponding to a field of view of 45°). The pixel lines extracted from the eight images are concatenated to give a panoramic view of the scene (see Figure 4). The concatenated image is traversed horizontally

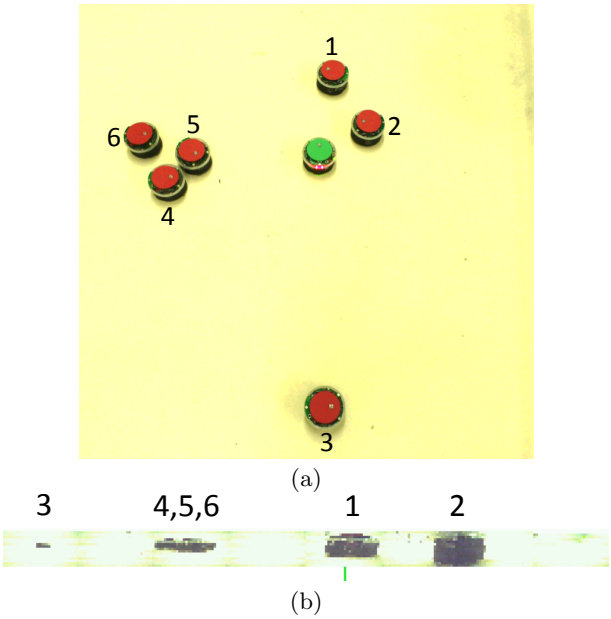


Figure 4: Image processing. (a) Overview of a scene with seven robots. (b) The corresponding concatenated image (here, with the original 15 pixel height) formed by the green robot as it takes eight images in one revolution. Note how the green robot sees the red robots 4, 5 and 6 as a single object that appears closer (see also Figure 5).

to scan for nearby robots. This is achieved by identifying blocks of dark pixels. Each block represents a perceived robot j. The angular position of that robot is estimated from the position of the block. Vector $\vec{v}_{ij,\text{repul}}$ points in the direction away from robot j. The distance to the robot, d_{ij}, is estimated from the width of the block. The amount of repulsion from a perceived robot j is proportional to the perceived amount of intersection. Formally,

$$||\vec{v}_{ij,\text{repul}}|| = \begin{cases} k\,(2r_i - d_{ij}) & d_{ij} < 2r_i; \\ 0 & d_{ij} \geq 2r_i, \end{cases} \tag{2}$$

where $k = 0.2$.

The total repulsion on robot i, $\vec{v}_{i,\text{repul}}$, is giving by summing the individual repulsion vectors for all blocks.

The vision based implementation differs from [12] in that two types of misperceptions can occur: (i) it is possible for several robots to be perceived as a single block of pixels [see Figure 5(a)]; (ii) it is possible for a robot to occlude one or more robots completely [see Figure 5(b)]. In order to compensate for these misperceptions, our repulsion mechanism places more emphasis on robots that are perceived to be close [see Equation (2)]. This is in contrast with the mechanism used in simulation [12], where the amount of repulsion is constant regardless of the distance to a perceived robot.

2.3 Experimental Setup

We use n to denote the number of robots in the swarm. Furthermore, we use m to denote the number of groups, and n_k to denote the number of robots in group k, $k \in \{1, 2, \ldots, m\}$. The robots in group k all have virtual radius $r^{(k)}$. Recall that r_i denotes the virtual radius of robot i.

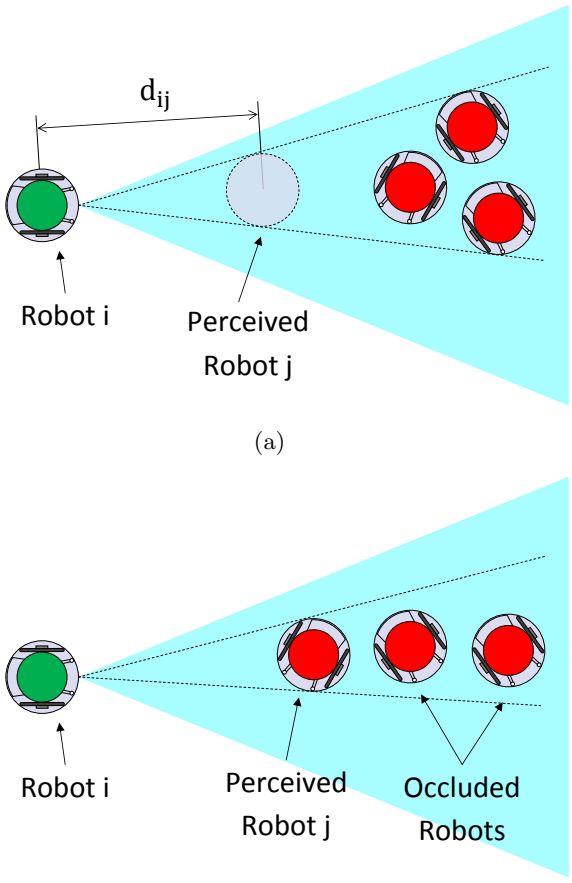


Figure 5: Possible misperceptions. (a) Robot i sees three overlapping robots as a single object, j. It incorrectly perceives a single robot at distance d_{ij}. (b) Robot i can not see the two robots occluded by robot j.

Thus, $r_i = r^{(k)}$, if robot i is in group k.

We consider a system with $n = 20$ robots and with $m = 2$ different groups. The virtual radius of robots from group k is chosen as follows:

$$r^{(k)} = ab^{k-1}, \tag{3}$$

where a is the size (in cm) of the smallest disk and b is the minimum size ratio between disks of different groups. We use $a = 8\,\text{cm}$ and $b \in \{1, 2, 3, 4, 5\}$.

Ideally, we expect the robots to organize into an annular structure, where the disks of radius $r^{(k)}$, $k \in \{1, 2, \ldots, m\}$, are fully contained within the area of the annulus formed by the concentric circles of radii $(k-1)g$ and kg in the center of the environment. Parameter g represents the "thickness" of the annulus and can be controlled by group size n [12].

An approximation of the ideal pattern can be obtained by choosing n_k as follows [12]:

$$n_k = \frac{\frac{2k-1}{(r^{(k)})^2}}{\sum_{j=1}^{m} \frac{2j-1}{(r^{(j)})^2}} n. \tag{4}$$

Table 1: Overview of configurations studied.

radius factor b	n_1	$r^{(1)}$	n_2	$r^{(2)}$
1.0	5	8.0 cm	15	8.0 cm
2.0	11	8.0 cm	9	16.0 cm
3.0	15	8.0 cm	5	24.0 cm
4.0	17	8.0 cm	3	32.0 cm
5.0	18	8.0 cm	2	40.0 cm

In our physical implementation, the robots moved in a square arena of sides 2.5 m. A light bulb was placed over the center of the arena, acting as the infrared radiation source, that is, the point of attraction.

The initial placement of the robots was done as follows: a square grid of 6×6 points was marked on the arena floor, centered around the light bulb, with all points being 20 cm apart. For each trial, 20 points were chosen randomly without replacement. Additionally, for each robot, the orientation was selected randomly from four possibilities: north, south, east and west.

Each trial was recorded from start to finish with an overhead camera system.

2.4 Performance Metric

To measure the quality of segregation, we calculate the *segregation error* (SE) as defined in [12]. Consider two robots i and j and let $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ denote their positions. Furthermore, let $\boldsymbol{o}$ denote the position of the 'center of gravity' in the same co-ordinate system, that is, the point to which all robots are attracted.

The pair of robots (i, j) contributes to the segregation error if one of the robots has a larger virtual radius *and* is closer to $\boldsymbol{o}$ than the other one. It does not contribute to the segregation error if either the robots have identical virtual radii, or if the robot with a smaller virtual radius is closer to $\boldsymbol{o}$ than the other one. Formally,

$$e_{ij} = \begin{cases} 1 & (r_i < r_j) \wedge (\|\boldsymbol{x}_i - \boldsymbol{o}\| \geq \|\boldsymbol{x}_j - \boldsymbol{o}\|)\,; \\ 1 & (r_i > r_j) \wedge (\|\boldsymbol{x}_i - \boldsymbol{o}\| \leq \|\boldsymbol{x}_j - \boldsymbol{o}\|)\,; \\ 0 & \text{otherwise.} \end{cases}$$

The segregation error is given by summing e_{ij} over all pairs of robots, and normalizing by (only) the number of errors possible. Formally,

$$\mathrm{SE} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} e_{ij}}{n^2 - \sum_{k=1}^{m} n_k^2}, \tag{5}$$

where $\mathrm{SE} \in [0, 1]$. Randomly placed robots will have a segregation error of 0.5 on average. An error of 1.0 is achieved if the robots are in an 'inverse Brazil nut' configuration, that is, if for all (i, j) s.t. $r_i < r_j$, $\|\boldsymbol{x}_i - \boldsymbol{o}\| \geq \|\boldsymbol{x}_j - \boldsymbol{o}\|$.

3. RESULTS

We considered $m = 2$ groups of robots. Robots of the first group represented disks of radius $r_1 = 8$ cm, whereas robots of the second group represented disks of radius $r_2 = 8b$ cm, $b \in \{1, 2, 3, 4, 5\}$. As reported in [19], the size ratio b is a critical variable—increasing it results in a decrease in the segregation error.

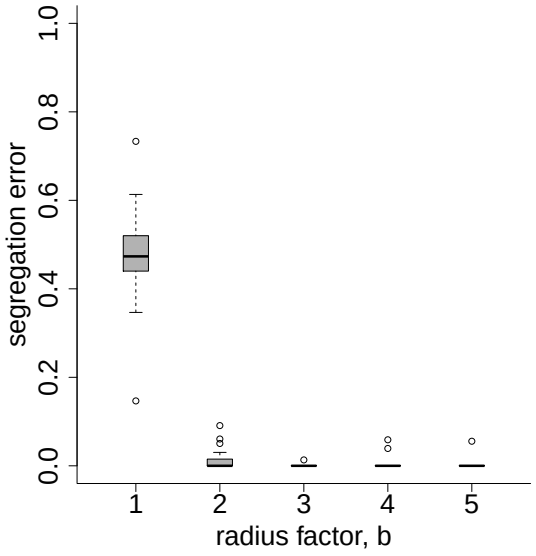


Figure 7: Box-and-whisker plot showing the segregation error observed in experimental trials with 20 e-puck robots for different radius factors (20 trials per radius factor). Each box comprises of observations ranging from the first to the third quartile. The median is indicated by a horizontal bar within the box. The whiskers extend to the farthest data points that are within 1.5 times the inter-quartile range. Outliers are indicated as circles.

For each value of b, we performed 20 trials with $n = 20$ robots each, that is, we ran 100 experimental trials in total. Every trial lasted for 20 minutes. Table 1 shows the number of robots in each group [see Equation (4)].

Figure 6 shows a sequence of snapshots taken during three typical trials with radius factor $b = 1$, 2 and 4.

3.1 Influence of Size Ratio on Segregation Error

Figure 7 shows a box-and-whisker plot [4] of the segregation errors for the different radius factors (b).

For $b = 1$, all e-pucks represented disks of identical size. Consequently, the segregation error (47.3%) was, on average, similar to the expected error for purely randomly distributed e-puck robots (50%). In no trial was perfect segregation observed.

For $b > 1$, the median segregation errors are all 0. The mean segregation errors are 1.31%, 0.07%, 0.49% and 0.28% for $b \in \{2, 3, 4, 5\}$, respectively. For $b = 2$, error free segregation was observed in 14 out of 20 trials. For $b \in \{3, 4, 5\}$, error free segregation was observed in 19, 18 and 19 of the 20 trials, respectively. That is, in these trials, all 20 e-pucks were spatially distributed as intended.

In 4 out of 60 trials for $b \in \{3, 4, 5\}$ the segregation was not error free. This was due to mechanical failures that caused robots to stop moving. For example, a robot became stuck on the arena floor, or lost contact with its battery.

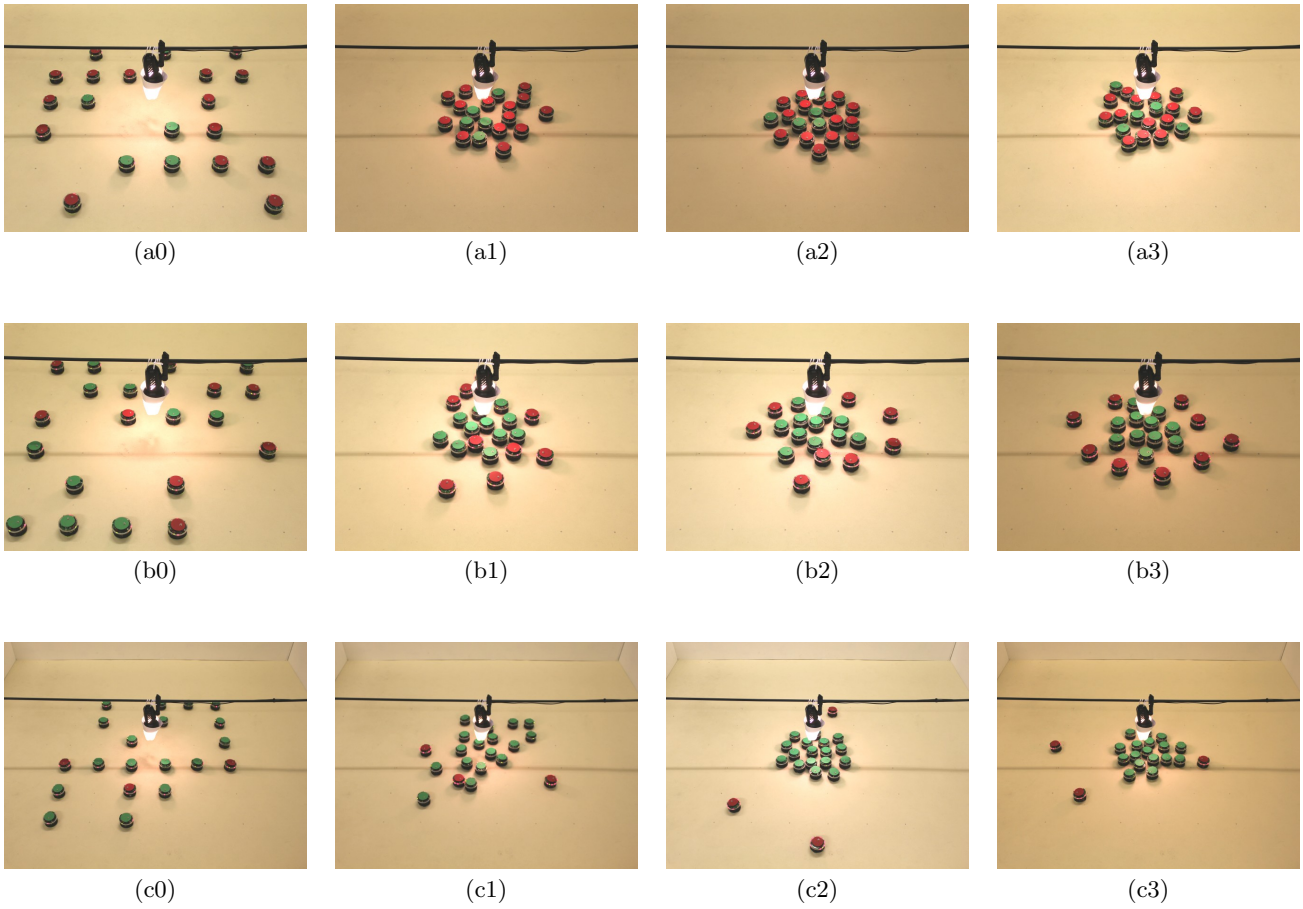


Figure 6: Sequences of snapshots taken during trials with radius factor b equal to 1 (top), 2 (center) and 4 (bottom). Robots with green markers represent disks of 8 cm radius. Robots with red markers represent disks of radius 8 cm (top), 16 cm (center) and 32 cm (bottom). The first and last images in each sequence (from left to right) show the initial and final configurations after 0 and 1200 s. The other two images show intermediate situations.

3.2 Influence of Size Ratio on Spatial Distribution

To understand better the effect of the size ratio (b), we analyzed the spatial distribution of robots of both groups. Figure 8 shows the distances of all robots from the center of the arena as observed at the end of the trial. The data is grouped according to the two groups of robots presenting disks of different sizes in addition to the radius factor used.

For $b = 1$, robots of both groups were similarly distributed in space. The mean distances from the center of 'smaller' robots (green marker) and 'larger' robots (red markers) were 16.9 cm and 17.5 cm, respectively.

As b increased, the distance between robots of different groups increased.

For robots representing small disks (of 8 cm radius), the mean distance from the center of the arena mainly depends on the number of disks of that size. The largest number of small disks was present for $b = 1$ (in this case, all 20 robots were identical). For $b \in \{2, 3, 4, 5\}$, the numbers were 11, 15, 17, and 18, respectively (see Table 1).

The mean distance of 'larger' robots from the center grew almost linearly with the radius factor, setting them spatially apart from the other group. This caused the segregation error to decrease.

3.3 Segregation Dynamics

Figure 9 shows the segregation error over time as observed in trials with radius factor $b = 4$. Initially, the segregation error rapidly decreased until it became zero after 3.5 mins in most of the trials.

4. CONCLUSIONS

In this paper, we studied spatial segregation in a swarm of physical robots. We described how to port an algorithm inspired by the Brazil nut effect from computer simulations [12] to the miniature mobile robot e-puck. The algorithm lets e-pucks mimic a mixture of disks under vibration.

We presented a series of experiments with 20 e-puck robots that confirm the efficacy of the algorithm. The e-pucks were programmed to simulate a system of two groups of disks. The desired target pattern was an annular structure around a common point of attraction, where the robots in each annulus represent disks of identical size. The percentage of incorrectly-ordered pairs of disks from different groups decreased as the size ratio of disks in different groups was in-

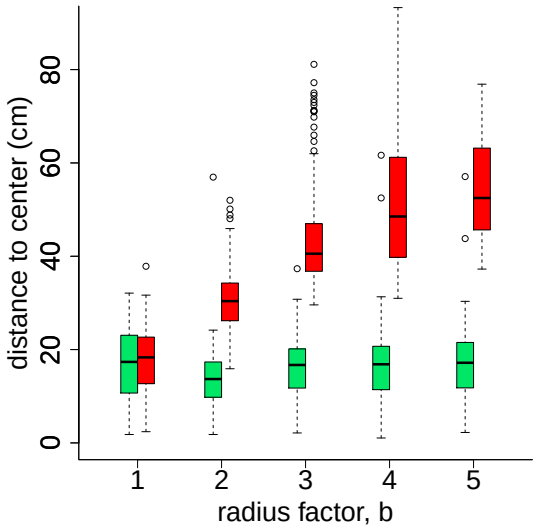


Figure 8: Box-and-whisker plot showing the distances of all robots from the center of the arena for groups of different radius factor (400 data points per radius factor). Green (light gray) boxes represent data from those robots that used the basic virtual radius, whereas red (dark gray) boxes represent data from those robots with the corresponding radius factor applied.

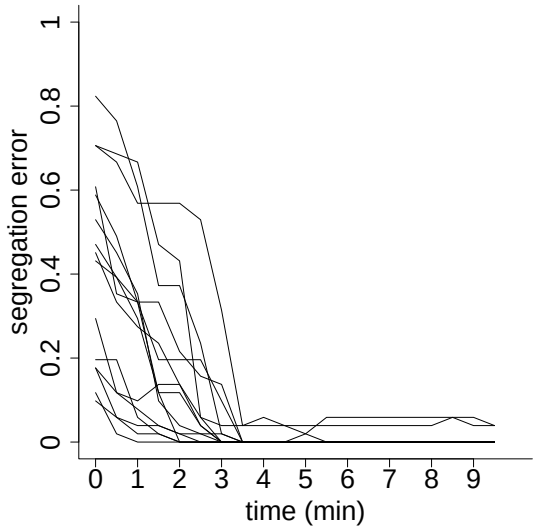


Figure 9: Segregation error over time for 15 experimental trials with 20 e-puck robots and radius factor $b = 4.0$. Data from the remaining five trials are not included because of some missing frames in the corresponding video recordings.

creased.[2] This percentage was, on average, below 0.5% for size ratios from 3.0 to 5.0. Moreover, for these size ratios, all segregation errors observed were due to mechanical failures that caused robots to stop moving. To the best of our knowledge, this is the first example of segregation in a swarm of physical robots with such a high level of accuracy.

The original algorithm in [12] assumed that every robot can instantly measure the relative position of all the robots in its vicinity. Here, we showed how this algorithm can be modified to allow for an implementation using directional vision. This implies that (i) robots have to revolve in order to obtain an omni-directional picture and (ii) the algorithm has to cope with misperceptions, for example, due to visual occlusion. We believe that the new algorithm is applicable to a wider range of robotic platforms when compared to the original algorithm. In principle, the new algorithm can be implemented on any wheeled robot with a camera or equivalent sensor to detect nearby robots. Note that the robot also needs to sense the angular position of a point of attraction in the environment (to emulate the effect of a gravitational pull). Here, we used a light bulb, which was perceived by the e-puck's infrared sensors. In principle, the light bulb could be perceived as well using the directional camera while the e-puck revolves to obtain the omni-directional picture.

The algorithm does not require the robots to communicate, nor does it require them to discriminate between each other. Therefore, the performance of the algorithm could possibly scale well with both the number of robots and the number of groups. Simulation results [12] support this claim; in these, the segregation error decreased exponentially with the size ratio and error free segregation was reported for 150 agents of three distinct groups.

In principle, the algorithm could form annular structures with an arbitrary number of nested layers as well as structures in three dimensions [9]. A present limitation, however, is that the robots' minimum sensing range could be required to increase exponentially with the number of layers.

5. ACKNOWLEDGMENTS

The research work disclosed in this publication is funded by the Marie Curie European Reintegration Grant within the 7$^{\text{th}}$ European Community Framework Programme (grant no. PERG07-GA-2010-267354).

M. Gauci acknowledges support by the Strategic Educational Pathways Scholarship (Malta). The scholarship is part-financed by the European Union – European Social Fund (ESF) under Operational Programme II – Cohesion Policy 2007–2013, "Empowering People for More Jobs and a Better Quality of Life".

The authors thank Andrew Hills for helpful comments on an earlier version of this paper.

6. REFERENCES

[1] M. Amos and O. Don. Swarm-based spatial sorting. *International Journal of Intelligent Computing and Cybernetics*, 1(3):454–473, 2008.

[2] R. C. Arkin. Motor schema-based mobile robot navigation. *Int. J. Robot. Res.*, 8(4):92–112, 1989.

[3] G. Barker and M. Grimson. The physics of muesli. *New Sci.*, 126(1718):37–40, 1990.

[2]Video recordings of the experimental trials can be found in the online supplementary material [7].

[4] R. A. Becker, J. M. Chambers, and A. R. Wilks. *The new S language. A programming environment for data analysis and graphics*. Chapman & Hall, London, 1988.

[5] N. Bowden, A. Terfort, J. Carbeck, and G. M. Whitesides. Self-assembly of mesoscale objects into ordered two-dimensional arrays. *Science*, 276(5310):233–235, 1997.

[6] S. Camazine, J.-L. Deneubourg, N. R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-organization in biological systems*. Princeton Univ. Press, Princeton, NJ, 2001.

[7] J. Chen, M. Gauci, M. J. Price, and R. Groß. Online supplementary material. `http://naturalrobotics.group.shef.ac.uk/supp/2012-001`, 2012.

[8] E. Şahin, T. Labella, V. Trianni, J.-L. Deneubourg, P. Rasse, D. Floreano, L. Gambardella, F. Mondada, S. Nolfi, and M. Dorigo. Swarm-bot: Pattern formation in a swarm of self-assembling mobile robots. In *Proc. of the 2002 IEEE Int. Conf. on Systems, Man and Cybernetics (SMC 2002)*, volume 4. IEEE Computer Society Press, Los Alamitos, CA, 2002.

[9] S. Foster and R. Groß. Forming nested 3D structures based on the Brazil nut effect. In *Proc. of the 12th Conf. Towards Autonomous Robotic Systems (TAROS 2011)*, volume 6856 of *Lecture Notes in Artificial Intelligence*, pages 394–395, Berlin, Germany, 2011. Springer-Verlag.

[10] N. R. Franks and A. B. Sendova-Franks. Brood sorting by ants: distributing the workload over the work-surface. *Behav. Ecol. Sociobiol.*, 30(2):109–123, 1992.

[11] R. Groß and M. Dorigo. Self-assembly at the macroscopic scale. *P. IEEE*, 96(9):1490–1508, 2008.

[12] R. Groß, S. Magnenat, and F. Mondada. Segregation in swarms of mobile robots based on the Brazil nut effect. In *Proc. of the 2009 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS 2009)*, pages 4349–4356. IEEE Computer Society Press, Los Alamitos, CA, 2009.

[13] M. Kumar, D. Garg, and V. Kumar. Segregation of heterogeneous units in a swarm of robotic agents. *IEEE T. Automat. Contr.*, 55(3):743–748, 2010.

[14] C. Melhuish, A. B. Sendova-Franks, S. Scholes, I. Horsfield, and F. Welsby. Ant-inspired sorting by robots: the importance of initial clustering. *J. R. Soc. Interface*, 3(7):235–242, 2006.

[15] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli. The e-puck, a robot designed for education in engineering. In *Proc. of the 9th Conf. on Autonomous Robot Systems and Competitions*, volume 1, pages 59–65. IPCB: Instituto Politécnico de Castelo Branco, 2009.

[16] A. M. T. Ngouabeu, S. Miyashita, R. M. Füchslin, K. Nakajima, M. Göldi, and R. Pfeifer. Self-organized segregation effect on self-assembling robots. In *Proc. of the 12th Int. Conf. on the Synthesis and Simulation of Living Systems (Artificial Life XII)*, pages 232–238. MIT Press, Cambridge, MA, 2010.

[17] A. Rosato, K. J. Strandburg, F. Prinz, and R. H. Swendsen. Why the Brazil nuts are on top: size segregation of particulate matter by shaking. *Phys. Rev. Lett.*, 58(10):1038–1040, 1987.

[18] T. C. Schelling. Models of segregation. *Am. Econ. Rev.*, 59(2):488–493, 1969.

[19] J. C. Williams and M. I. Khan. The mixing and segregation of particulate solids of different particle size. *The Chemical Engineer*, 269:19–25, 1973.

[20] M. Wilson, C. Melhuish, A. B. Sendova-Franks, and S. Scholes. Algorithms for building annular structures with minimalist robots inspired by brood sorting in ant colonies. *Auton. Robot.*, 17(2–3):115–136, 2004.

Model-Driven Behavior Specification for Robotic Teams

Alexandros Paraschos*
IAS Lab
TU-Darmstadt
Darmstadt, 64287, Germany
paraschos@ias.tu-darmstadt.de

Nikolaos I. Spanoudakis
Department of Sciences
Technical University of Crete
Chania, 73100, Greece
nikos@science.tuc.gr

Michail G. Lagoudakis
Department of ECE
Technical University of Crete
Chania, 73100, Greece
lagoudakis@ece.tuc.gr

ABSTRACT

Modern model-driven engineering and Agent-Oriented Software Engineering (AOSE) methods are rarely utilized in developing robotic software. In this paper, we show how a Model-Driven AOSE methodology can be used for specifying the behavior of multi-robot teams. Specifically, the Agent Systems Engineering Methodology (ASEME) was used for developing the software that realizes the behavior of a physical robot team competing in the Standard Platform League of the RoboCup competition (the robot soccer world cup). The team consists of four humanoid robots, which play soccer autonomously in real time utilizing the on-board sensing, processing, and actuating capabilities, while communicating and coordinating with each other in order to achieve their common goal of winning the game. Our work focuses on the challenges of coordinating the base functionalities (object recognition, localization, motion skills) within each robot (intra-agent control) and coordinating the activities of the robots towards a desired team behavior (inter-agent control). We discuss the difficulties we faced and present the solutions we gave to a number of practical issues, which, in our view, are inherent in applying any AOSE methodology to robotics. We demonstrate the added value of using an AOSE methodology in the development of robotic systems, as ASEME allowed for a platform-independent team behavior specification, automated a large part of the code generation process, and reduced the total development time.

Categories and Subject Descriptors

D.2.10 [**Software Engineering**]: Design—*Methodologies*; I.2.9 [**Artificial Intelligence**]: Robotics—*Commercial robots and applications*

General Terms

Design

Keywords

Agent-Oriented Software Engineering, Robotic Software Development, Intra-Agent Control, Model-Driven Engineering

*Work performed while at the Technical University of Crete.

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

1. INTRODUCTION

The Model-Driven Engineering (MDE) paradigm has gained popularity among software developers and a number of methodologies, models, and tools have been developed to facilitate task decomposition, enable software reusability, minimize coding mistakes, and allow for inexpensive software maintenance. Even though some of this technology has been extended to cover the needs of agent-oriented software development, it is rarely exploited to address the needs of robotic software. Indeed, the real-time constraints and the concurrency of device operation in robotics typically impose a low-level of coding, whereas the potential of programming high-level behaviors that exploit the lower-level functionalities at a meta-level of coding remains largely unexplored.

Agile processes address several modern software development needs, such as the need for coping with continuously changing requirements, the need for continuous evaluation, and the need for less bureaucracy related to the extensive production of models that few people (only the developers) can read [8]. Thus, agile processes are very useful for projects, such as the development of a RoboCup team, whereby an autonomous robot team competes against another and the behavior of the robots may need to change between games to cope with skilled opponents.

Given that robots capture the inherent properties of agents (autonomy, social ability, reactivity, proactiveness), in this paper, we show how an Agent-Oriented Software Engineering (AOSE) methodology can be used for specifying the behavior of multi-robot teams encompassing the model-driven and agile characteristics described above. More specifically, we focus on the Agent Systems Engineering Methodology (ASEME) [21] and the domain of robotic soccer. According to Schlegel et al. [17], software engineering for robotics is different in some important aspects from software engineering for other, even related, areas, such as distributed and real-time systems. Thus, in our work we had to adapt the ASEME process to accommodate the needs of robotic software development. In this context, we defined a generic transformation tool (IAC2Monas) for instantiating the statechart models of ASEME (the platform-independent models [11]) to our Monas robot software architecture [14] for integration with implemented functionalities and execution on our robots; this coupling provided automatic code generation and execution on a generic multi-threaded statechart engine along with the Monas software modules. As a result, instead of specifying complex team behavior using hundreds of lines of conventional code, the developer can now accomplish this task using an intuitive graphical representation

with advanced control modes. Our work demonstrates the added value of using the ASEME methodology in robotics, as it allowed for a platform-independent team behavior specification, automated a large part of the code generation process, eliminated common coding mistakes, and reduced the total development time.

2. BACKGROUND AND MOTIVATION

It is common practice for roboticists to specify a robot's behavior using conventional procedural code, whereby a complex arrangement of conditional statements determines what the robot is supposed to do in each condition. A higher-level practice is to specify a robot's behavior using the formalism of Finite State Automata (FSA) whose graphical representation with nodes (states) and edges (transitions) offers a more intuitive way of synthesizing the desired behavior. However, as robots become more complex and employ the computing power of modern processors, their programming also becomes more demanding, requiring concurrent and threaded code to support efficient implementations of advanced operations, such as machine learning and signal processing algorithms. Thus, there is a clear need for modern software engineering methods in developing robotic software.

Statecharts [6], a formal model familiar to software developers, have been widely used for specifying agent plans, even in the RoboCup domain [12, 13]. Murray [12], in particular, proposes the use of extended statecharts (with synch states for synchronizing the actions of different agents) for defining the behavior of RoboCup simulation players. This work is also supported by an editing tool (StatEdit). Both proposals [12, 13] support semi-automatic code generation for Robolog, a robot programming language based on Prolog. These approaches have been used only in RoboCup simulation leagues and it is not clear how they could be adapted for use on real robots. In that case, base functionalities, such as perception and locomotion, which are provided freely in the simulation leagues, will have to be inserted into the statechart. It is not straightforward how this can be done, when a procedural programming language, such as Python or C++, is used for implementing these functionalities.

Recent developments in Multi-Agent Systems (MAS) have demonstrated that high-level approaches, such as the Extensible Agent Behavior Specification Language (Xabsl) [15] and Petri Net Plans (PNPs) [25], can be utilized for the behavioral modeling of robots. Despite their different formal models, hierarchical FSAs for Xabsl and Petri Nets for PNPs, both approaches offer hierarchical decomposition of complex behaviors, concurrent action support within their formalism, and multi-robot coordination. Although, PNPs have a more compact representation than FSAs, they still require more semantics than statecharts. Integration with state-of-the-art frameworks (B-Human [16] for Xabsl and OpenRDK [1] for PNPs) provides a threaded, low-latency environment for efficient runtime execution. Analysis and validation of the designed models can be done using standard tools, due to the use of formal and widely-used representations. Both approaches have been employed successfully in the RoboCup competition. However, both of them model only behavioral, but not functional, aspects of the system. Moreover, inter-agent coordination protocols cannot be integrated directly into their formal models.

In their work, De Loach et al. [2] apply the Multi-agent Systems Engineering (MaSE) methodology for designing teams of cooperating robots. They use a top-down approach, starting from system goals and gradually refining them to simpler goals. They use sequence diagrams for designing interaction protocols and independent instances of finite state machines (called concurrent task diagrams) for designing the behavior of each identified agent role. However, their approach is quite limiting, as it allows only for bilateral conversations, thus favoring centralized coordination schemes. Moreover, the lack of hierarchical structure in the agent plans leads to flat, large, and complex representations.

Other authors, such as Gascuena and Fernandez-Caballero [5], used the Prometheus methodology [23] to specify the behavior of a robot. Prometheus provides specific diagrams for depicting the agent roles, their resources, and exchanged messages with other roles. It uses AUML Agent Interaction Protocol (AIP) diagrams (extended UML[1] sequence diagrams) for specifying agent interactions. However, the seven different types of diagrams they propose are always constructed manually anew. The support for implementation, testing, and debugging of Prometheus models is limited and available only for the JACK agent platform. Finally, they do not address the multi-tasking issue on a single robot, but rather identify each component/task as a distinct agent. Nevertheless, in practice a lot of information coming from sensors and other modules accomplishing specific tasks need to be processed concurrently and the timing between these tasks is critical. Finally, AUML agent coordination protocols are not integrated seamlessly in agent plans.

A method coming from the MDE community is presented by Schlegel et al. [17]. The authors argue for switching the traditional code-driven robotic software development to a model-driven one. They define strict interfaces for wrapping existing components and then utilize a statechart-based approach for specifying the behavior of robots. Then, they use MDE techniques for transforming the platform-independent model they define to executable code. Their approach is a significant step towards model-based software engineering for robots, lacking mostly in the multi-agent aspect, as there is no catering for agent interaction protocols definition.

The Agent Systems Engineering Methodology (ASEME) [21] fills this particular gap. ASEME supports a modular agent design approach and introduces the concepts of intra- and inter- agent control. The former defines the agent's behavior by coordinating the different modules that implement its own capabilities, while the latter defines the protocols that govern the coordination of the society of the agents. ASEME applies an MDE approach to multi-agent systems development, so that the models of a previous development phase can be transformed to models of the next phase. The transition from one phase to another is assisted by automatic model transformation leading from requirements to computer programs. The ASEME platform-independent model, which is the output of the design phase, is a statechart that can be instantiated in a number of platforms using existing Computer-Aided System Engineering (CASE) tools.

ASEME specifies three levels of abstraction for each phase of the software development process. The first is the *societal level*, in which the whole multi-agent system functionality is modeled. Then, the *agent level* zooms on each member of the society, i.e. the individual agent. Finally, the details that compose each of the agent's parts are defined in the

[1]The Unified Modeling Language (UML) is a standardized object-oriented modeling language: www.uml.org

Development Phase	Levels of Abstraction		
	Society Level	Agent Level	Capability Level
Requirements Analysis *AMOLA Models*	Actors `System Actors Goals (SAG): Actors`	Goals `SAG: Goals`	Requirements `SAG: Requirements per goal`
Analysis *AMOLA Models*	Roles and Protocols `System Use Cases (SUC), Agent Interaction Protocols (AIP)`	Capabilities `SUC, System Roles Model (SRM)`	Functionalities `SRM: Activities and Functionalities`
Design *AMOLA Models*	Society Control `Inter-Agent Control (EAC)`	Agent Control `Intra-Agent Control (IAC)`	Components
Implementation	Platform management code	Agent code	Capabilities code
Verification	Protocols testing	Agent testing	Component testing
Optimization	Number of instantiated agents	Agent resources	Code optimization

Figure 1: ASEME phases and AMOLA products.

capability level. The concept of *capability* is defined as the ability of an agent to achieve specific tasks that require the use of one or more *functionalities*. The latter refers to the technical solution(s) to a given class of tasks. Moreover, capabilities are decomposed to simple *activities*, each of which corresponds to exactly one functionality. Thus, an activity corresponds to the instantiation of a specific technique for dealing with a particular task (a unique characteristic compared to the other statechart-based approaches). ASEME is mainly concerned with the first two abstraction levels, assuming that development in the capability level can be achieved using classical (or even technology-specific) software engineering techniques.

In Figure 1, the ASEME phases, the different levels of abstraction, and the models related to each one of them are presented. ASEME uses the models of the Agent Modeling Language (AMOLA) [19]. The AMOLA metamodels have been formally defined using the Eclipse Modeling Framework of the Eclipse Modeling Project[2]. Eclipse technology has been employed for developing model transformations and graphical editing tools for both models and processes[3].

3. ROBOCUP, NAO, AND SPL

In its short history, the RoboCup competition [10] (robot soccer world cup) has grown to a well-established annual event bringing together the best robotics researchers internationally. To succeed in playing soccer autonomously, the core problems of artificial intelligence and robotics (perception, cognition, action, coordination) must be addressed simultaneously under real-time constraints. The proposed solutions are tested through soccer games in various leagues. A key aspect of most RoboCup leagues is the multi-agent environment. The robots in each team cannot simply act as individuals; they must focus on teamwork in order to cope effectively with an unknown opponent team and such teamwork requires coordination.

The Standard Platform League (SPL)[4] is among the most popular leagues, featuring four humanoid Aldebaran Nao robot players in each team. The Nao is a 58cm, 4.3Kg humanoid robot developed by Aldebaran Robotics in Paris, France. It is equipped with an x86 AMD Geode processor at 500 MHz, 256 MB SDRAM, 2 GB flash disk, two color cameras, two ultrasound sensors, an inertial unit (2 gyroscopes and 3 accelerometers), an array of force sensitive resistors on each foot, encoders on all servos, and a total of 21 degrees of freedom (4 in each arm, 5 in each leg, 2 in the head, and 1 in the pelvis). SPL games take place in a $4m \times 6m$ field marked with white lines on a green carpet with two colored (skyblue and yellow) goals. Each game consists of two 10-minutes halves and teams switch sides at halftime. There are several rules enforced by human referees during the game.

In SPL, all teams use the same robotic hardware and differ only in terms of their software. Therefore, research efforts focus on developing more efficient algorithms and techniques for visual perception, active localization, omnidirectional motion, skill learning, individual robot behavior specification, and team coordination strategies. This paper focuses on the last two challenges.

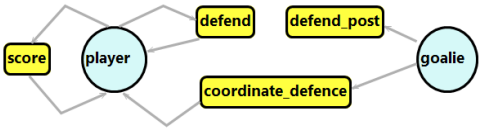


Figure 2: The System Actors Goals (SAG) model.

4. SOFTWARE ENGINEERING PROCESS

In this section, we describe the proposed model-based agent-oriented software engineering process in a step-by-step manner following the principles of AMOLA and using the RoboCup domain as our case problem.

4.1 Requirements Analysis Phase

In the requirements analysis phase, AMOLA defines the *System Actors and Goals* (SAG) model, containing the main actors in the system and their goals. For the Robocup domain, the actors are the players and the goalie of the team (see Figure 2). The player aims to score and defend, both goals depending also on the other players. The goalie aims to defend its post (individual goal), but also to coordinate the defense (depending on the players).

4.2 Analysis Phase

In the analysis phase, AMOLA proposes the *System Use Cases* (SUC) model, where the different activities that realize the agent capabilities are defined in a top-down decomposition process, the *Agent Interaction Protocol* (AIP) model, which specifies the coordination between agents, and, finally, the *System Roles Model* (SRM), through which the previously-defined activities are integrated to define the dynamic behavior of the roles of the agents. Initially, the SAG model from the previous phase is transformed to the SUC model (see Figure 3). The SAG goals are transformed to SUC use cases and the SAG actors to SUC roles. The modeler optionally adds roles and ≪includes≫ use cases. The goals transformed to use cases form the roles' capabilities.

In Figure 3, the *score* capability has been decomposed to simpler use cases using the ≪includes≫ relation. Thus, for scoring, the player can kick the ball towards the goal (*kick ball* use case) or participate in the coordinated *attack* use

[2]The Eclipse Modeling Project provides a unified set of modeling frameworks, tooling, and standards implementations: www.eclipse.org/modeling
[3]The AMOLA metamodels and ASEME transformation tools are freely available from: www.amcl.tuc.gr/aseme
[4]SPL Web Site: www.tzi.de/spl

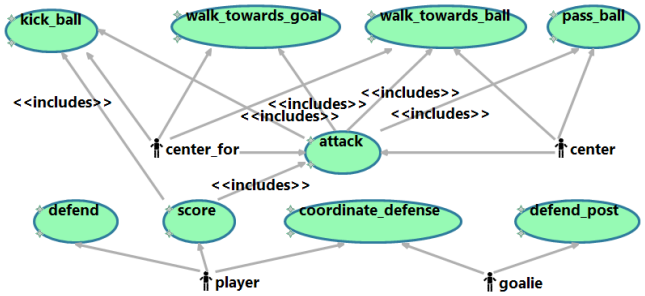


Figure 3: The System Use Cases (SUC) model.

Table 1: The AIP model for the *attack* protocol.

Participants	*center*	*center_for*
Engagement Rules	no robot has control of the ball ***and*** *center* is the robot closest to the ball ***and*** *center_for* is the robot farthest from the ball	
Outcomes	the *center_for* shoots to goal ***or*** an opponent takes control of the ball ***or*** the ball goes out of bounds	
Process	WalkTowardsBall. [passBall]	WalkTowardsGoal. [WalkTowardsBall. [kickBall]]

case. Note that the *attack* use case has also been associated with two new roles, the *center* and *center for*, which are connected to new use cases, decomposing further the *attack* use case. Thus, the *center* walks towards the ball and passes it to the *center for*, while the *center for* walks towards the opponent's goal post to receive the pass, then walks towards the ball and kicks it. The SUC model does not specify the order in which use cases are employed by the roles. The AIP and SRM models fill exactly this gap; the former specifies how to coordinate the cooperative roles' activities and the latter how to coordinate the individual role's activities.

Thus, as the human soccer team coach sketches the players' movements for a coordinated team action in real soccer, the robotic team coach uses the AIP model to sketch the robotic team's coordinated activities. The AIP model lists the participants along with the preconditions and postconditions in free text format. The process of each participant, however, is described formally using liveness formulas. Liveness formulas connect activities using the Gaia operators [24]. Briefly, $A.B$ means that activity B is executed after activity A, A^{ω} means that A is executed continuously (it restarts as soon as it finishes), $A|B$ means that either A or B is executed, $A||B$ means that A and B are executed in parallel, and $[A]$ means that A is optional.

The *attack* protocol with two participant roles (i.e. *center* and *center for*) is presented in Table 1. The rule for engaging in these roles is depicted in the second row, followed by the expected outcomes in the third row, and the process for each role defined using liveness formulas in the fourth row. In particular, the player closer to the ball becomes the *center* and the other players become *center for*. The *center* is expected to approach the ball and pass it to a *center for* who, in turn, is expected to be near the opponent's goal post to receive the pass and shoot to score. The protocol may terminate early, if an opponent takes control of the ball or the ball goes out of bounds.

The System Roles Model (SRM) defines each concrete role (corresponding to a SAG actor) by specifying the protocols in which the role participates and liveness formulas defining its dynamic behavior including the relevant process parts of the AIP model. Figure 4 shows the SRM for the player role. Note that this role can participate in the *attack* protocol either as a *center* or as a *center for*. While in the AIP model process part the activities are abstractly defined, in the SRM liveness formula all activities are connected to specific functionalities of the robot. The identified functionalities in our case are the following:

- *Sensors*, for collecting and filtering all data from the robot sensors (accelerometers, buttons, bumpers, etc.),
- *RobotController*, for listening to external information about the game state coming from the game controller,
- *LedHandler*, for managing the operation of the colored LEDs of the robot (eyes, ears, buttons),
- *MotionController*, for scheduling and executing motion commands (walk, kick, stand-up, special actions, etc.),
- *Vision*, for detecting the ball and the goals in the camera image and estimating their distance and bearing,
- *Localization*, for estimating the position and orientation of the robot and the ball in the field,
- *ObstacleAvoidance*, for planning obstacle-free paths in a local polar map using ultrasonic range measurements,
- *HeadHandler*, for managing the movements of the robot head and, thus, the camera (scanning, tracking, etc.).

The self-explained activities named *Stand, CalibrateCamera, CheckForBallObservation, ScanForBall, TrackBall, WalkTowardsBall, KickBall, PassBall, WalkTowardsOpponentGoal* are provided either directly by the above functionalities or by combining information coming from some of them (for example, *Vision* with *ObstacleAvoidance* and *MotionController* to realize *WalkTowardsBall*). Similarly to the work of Schlegel et al. [17], all functionalities are wrapped with standard interfaces. In our Monas architecture they are defined through XML configuration files.

4.3 Design Phase

In the design phase, AMOLA defines the *intEr-Agent Control* (EAC) model and the *Intra-Agent Control* (IAC) model, which are based on the formalism of *statecharts* and define both the functional and behavioral aspects [6] of the multi-agent system. The ASEME SRM2IAC tool is used to transform the process formulas of an AIP model protocol to an EAC model and the liveness formulas of an SRM role to an IAC model. The EAC and IAC models are statecharts, where the developer can insert events, conditions, and actions in the transition expressions, thus controlling each role's process either for satisfying the needs of a protocol (in the EAC model) or for coordinating the agent's capabilities (in the IAC model). There are six types of *states* in a statechart [6]:

- *start*, showing where execution starts
- *end*, showing where execution stops
- *or*, having sub-states (of any kind) related by "exclusive-or", i.e. only one is executed at any given time

Role: player
Protocols: attack: center, attack: center_for
Liveness:
player = Sensors$^{\omega}$ || RobotController$^{\omega}$ || LedHandler$^{\omega}$ || MotionController$^{\omega}$ || (initialize . activate)
initialize = Stand . CalibrateCamera
activate = Vision$^{\omega}$ || Localization$^{\omega}$ || ObstacleAvoidance$^{\omega}$ || HeadHandler$^{\omega}$ || decision$^{\omega}$
decision = CheckForBallObservation . (ScanForBall | action)
action = TrackBall || (WalkTowardsBall | KickBall | center | center_for)
center = WalkTowardsBall . [PassBall]
center_for = WalkTowardsOpponentGoal . [WalkTowardsBall . [KickBall]]

Figure 4: The SRM model for the *player* (participating as *center* or *center_for* in the *attack* protocol).

- *and*, having *or*-states as sub-states related by "and", i.e. all of them are executed concurrently
- *basic*, having no sub-states, representing an activity
- *condition*, offering only conditional transitions (also known as *OR-connector* or *conditional transition*)

The state at the highest level (the one with no parent state) is called the *root*. Each *transition* from one state (source) to another (target) is labeled by an *expression*, whose general syntax obeys the pattern `e[c]/a`, where e is the event that triggers the transition; c is a condition that must be satisfied for the transition to be taken, when event e occurs; and a is an action that takes place, when the transition is taken. All elements of the transition expression are optional. The *scope* of a transition is the lowest level *or*-state, which is a common ancestor of both the source and target states. When a transition occurs all states in its scope are exited and the target states are entered.

Having defined the statechart, as it is used in AMOLA [20], it is now possible to proceed to the definition of the inter-agent control (EAC) model. The EAC is a statechart that contains an initial (*start*) state, an *and*-state named after the protocol, and a final (*end*) state. The *and*-state contains as many *or*-states as the protocol roles, named after these roles. One transition connects the *start*-state to the *and*-state and another transition the *and*-state to the *end*-state. Transitions can be triggered by a timeout event or by the completion of the executed state activity. Thus, for the *attack* protocol, since the two participating roles operate simultaneously in parallel, the SRM2IAC tool transforms the following formula along with the process part of the AIP model protocol into a statechart:

```
action = center || center_for
```

The result of the automatic model transformation is depicted graphically in the form of an ordered rooted tree (Figure 5) that defines the statechart. In the fragment of the statechart shown in Figure 5, the reader can see the *center* role. The nodes of the tree (rounded rectangles) define the states (gray lines point to parent nodes in the tree structure). Each node includes the state name and the state type. Labeling the nodes properly helps the modeler identify the position of a node within the tree. For example, the $A.1$ label means that the node labeled with A is the parent of the node labeled with $A.1$. Nodes without a name coming from the formula, e.g. *start* nodes, are named after their label. Each *or*-state includes a *start*-state and, usually an *end*-state (except in the cases where a state loops infinitely to its self, thus no end state is needed) to note where execution starts and where it stops. The modeler can now define transition expressions for all the transitions (depicted with red/dark lines) using the grammar defined in Figure 6 in EBNF format [9]. In Figure 5, the modeler has just defined a condition for the transition having as source the *condition*-state at the bottom of the figure and target the *basic* state to its right named *passBall*. It checks if the ball is within an angle of 10 degrees from the current orientation of the robot's torso and within a distance of 6 cm from the center of its feet, i.e. the robot can kick the ball to pass it.

The intra-agent control (IAC) model is also initiated by the SRM2IAC tool for each role. Again, the modeler must define the transition expressions and the variables contained in these expressions. By convention, the user should not define new transitions, although the statechart formalism allows for transitions between any two states, because the resulting IAC model will no longer represent the process defined by the liveness formulas. Moreover, the modeler must ensure that the branches of the statechart that come from EAC models are transferred unchanged in the statechart (their transition expressions must not change). Only in this way are the protocols guaranteed to be executed as planned. The RoboCup *player*'s statechart (IAC model) is shown in Figure 7, with the zoom window focusing on the *center* part of the *attack* protocol. Notice that the example condition introduced earlier appears unchanged at the correct place.

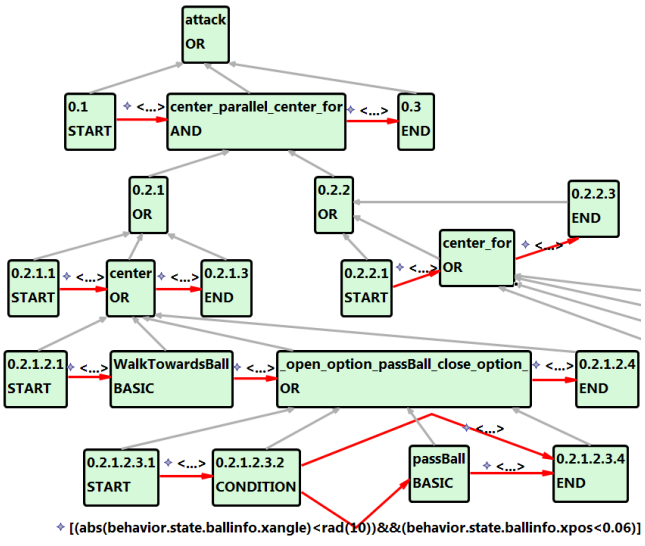


Figure 5: The EAC model for the *attack* protocol.

```
transitionExpression = [ event ] [ '[' condition ']' ] [ '/' actions ]
event = string
condition = expr | expr (compOp | logicOp) condition
          | '(' condition ')' | notOp condition
actions = action | action ';' actions
action = expr | variable '=' expr | 'read_messages'
       | 'write_messages' '.' topic '.' commType '.' msgType
expr = varVal | function '(' args ')'
function = string
args = varVal | varVal ',' args
varVal = variable | value
value = constant | stringLiteral
compOp = '<' | '<=' | '>' | '>=' | '==' | '!='
logicOp = '&&' | '||'
notOp = '!'
variable = host '.' topic '.' commType '.' msgType '.' member
         | topic '.' commType '.' msgType '.' member
commType = 'signal' | 'state' | 'data'
host = string
topic = string
msgType = string
member = string
stringLiteral = '"' string '"'
string = letter (letter | digit)*
letter = 'A'..'Z' | 'a'..'z' | '_'
digit = '0'..'9'
constant = [ '+' | '-' ] digit digit* [ '.' digit digit* ]
```

Figure 6: The transition expression grammar.

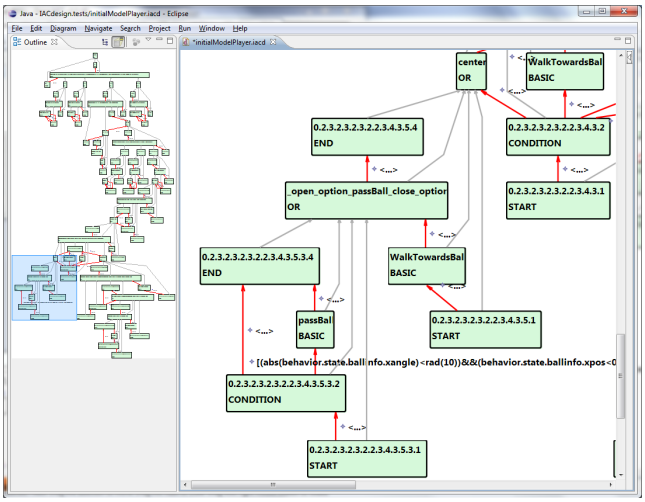


Figure 7: The IAC model for the RoboCup *player*.

4.4 Implementation Phase

To facilitate the code generation process, we built the IAC2Monas transformation tool [14], which translates the IAC model automatically to C++ source code adhering to the Monas architecture. IAC2Monas is a model-to-text (M2T) transformation tool. Thus, the platform-independent model (IAC) is transformed to a platform-specific model (code), which is subsequently cross-compiled to produce the executable for the robot. In order to build this tool we used the Xpand language offered by the Eclipse Modeling Project following the practice proposed by ASEME for the IAC2JADE transformation [21]. Xpand is used to define the templates for the required C++ classes, which are instantiated using information from the IAC metamodel elements, and is integrated with Xtend to handle the instantiation of complex expressions. An extract from the automatically generated statechart definition file (by the IAC2Monas tool) is presented in Figure 8. The reader can get an idea of how *or*-states, *basic* states, and *condition* states are defined in C++. The generation of a node (state) requires a name and the parent node as arguments, with the exception of the `Statechart` class, which appears only at the top of the hierarchy.

```
#include "AttackerPlan.h"

namespace { StatechartRegistrar<AttackerPlan>::
    Type temp("AttackerPlan"); }

AttackerPlan::AttackerPlan(Communicator* com) {

  _stc = new Statechart ( "Player", com );
  Statechart* Node0 = _stc;
  _states.push_back( Node0 );

  OrState* Node0212 = new OrState
    ( "RobotController_forever_", Node021 );
  _states.push_back( Node0212 );

  IActivity* ActivI02122 = ActivityFactory::
    Instance()->CreateObject( "RobotController" );
  _activities.push_back( ActivI02122 );
  BasicState* Node02122 = new BasicState
    ("RobotController",Node0212,ActivI02122);
  _states.push_back( Node02122 );

  ICondition* CondI02TO03 = new TrCond02TO03;
  _conditions.push_back( CondI02TO03 );

  _transitions.push_back( new TransitionSegment
    <State,State>(Node02, Node03, CondI02TO03) );
```

Figure 8: An extract of the auto-generated code.

To support this phase, we had to develop two important software libraries: (a) the communication framework both for inter-agent (i.e. between different agents) and intra-agent (i.e. between activities on a single agent) communication and (b) the statechart engine for executing statecharts.

Our communication framework [22] is based on the publish/subscribe messaging pattern [4] and supports multiple ways of communication, including point-to-point and multicast connections. The information that needs to be communicated between nodes (agents or activities) is formed as messages, tagged with appropriate topics, and relayed through a message queue for delivery. We used Google Protocol Buffers[5] to facilitate the serialization of data and the structural definition of the messages. Additionally, the blackboard paradigm [7] is utilized to provide efficient access to shared information stored locally at each node and is extended to support history queries and a mechanism that controls the information updates.

Our statechart engine [14] was built on top of existing open-source projects. Its main distinguishing characteristic

[5]Protocol Buffers are Google's language- and platform-independent, extensible mechanism for serializing structured data. http://code.google.com/apis/protocolbuffers

from other frameworks, e.g. UML and Boost[6], is the multi-threaded statechart execution that provides the required concurrency and meets the real-time requirements of the activities on each robot.

These libraries are linked to the automatically generated code at compilation time. A blackboard is instantiated (a) for each agent and (b) for each *or*-state that is a substate of an *and*-state. This way, the shared information is distributed, not only among network nodes (agents), but also among the concurrently executed parts of each agent. The latter allows for a significant increase in run-time performance, as it eliminates starvation and producer/consumer problems. The presented models do not include explicit communication activities, because coordination occurs through the `read_messages` and `write_messages` actions. These actions allow the use of shared information from the blackboards as variables in the transition expressions (see the grammar rules for *action* and *variable* in Figure 6).

5. EMPIRICAL EVALUATION

To empirically evaluate our approach, we compared it to our previous practice, i.e. using Aldebaran's middleware for Nao (NaoQi) which provides, besides the API, a platform for modular software development and a thread-safe mechanism for communication. As a proof of concept, a student familiar with both development methods was asked to develop the same behavior for the RoboCup team. The empirical results in terms of some development performance metrics are shown in Table 2. *Run-Time Performance* refers to the system load average over 5 minutes of execution. Values greater than one indicate CPU overload. The NaoQi-based agent had inferior performance, due to system overload caused by the vast amount of exchanged information between the modules. Writing C++ code had its impact on *Total Development Time*, which was considerably higher in NaoQi, whereas the statechart graphical editing tool allowed for quicker development. The ASEME-based agent consisted of more *Lines of Source Code*, however the vast majority of them were *Auto-Generated*. The NaoQi-based agent required the maintenance of several *State Variables* to indicate the current state of the agent, whereas in the ASEME-based agent it was represented explicitly in the statechart. The advantages of the ASEME approach were also reflected on the *Debugging Phase*, where NaoQi exhibited increased *Debugging Time* with a larger *Number of Bugs*.

Our experience from our RoboCup team[7] indicates that the model-based ASEME methodology with the automated transformation tools (SRM2IAC, IAC2Monas) is advantageous over our previous practice. New students familiarize themselves with robot team behavior specification in significantly less time. New ideas on team behavior can be quickly prototyped and existing behaviors can be easily explained. ASEME proves itself in behavior update or modification, which rarely involves the introduction of new functionality and typically amounts to changes in the agent's process and team protocols. This feature turned out to be extremely useful in the RoboCup 2011 competition, where we were able to modify our team behavior even at half-times or during timeouts. ASEME was one of this year's innovations that contributed to a significantly better team performance in the SPL games of RoboCup 2011 compared to RoboCup 2010 and led to winning the second place in the SPL Open Challenge Competition. The reader may watch our robot players in action at: www.kouretes.gr/aamas2012.mp4.

Table 2: Comparison of development methods.

Metric / Concept	NaoQi	ASEME
Run-Time Performance (load)	1.3	0.8
Development Phase		
Total Development Time	8 hours	5 hours
Lines of Source Code	490	826
Auto-Generated Lines	N/A	805
Number of State Variables	18	N/A
Debugging Phase		
Total Debugging Time	12 hours	5 hours
Number of Bugs	16	4

6. DISCUSSION

We faced several challenges in applying the ASEME AOSE methodology to multi-robot behavior specification, which, in our view, are inherent in this process and every AOSE practitioner will face in a similar endeavor.

Firstly, most AOSE methodologies take for granted that the agents communicate and coordinate through message passing. This does not always hold in robotic applications, where coordination can be based on diverse communication means, such as inter-agent messages, blackboards, or even sensory information. Even though ASEME defines interaction protocols based on the activities of the participants, the original statechart transition expression language [18] assumed that a FIPA[8]-like communication language would be used for message exchange. This is not true for most multi-robot applications, where the real-time constraints forbid the use of Java, on which the most successful agent platforms and those that comply to FIPA are based. The use of the publish/subscribe communication framework and the blackboard paradigm for local storage, forced us to modify the transition expression language.

Additionally, the transformation of the platform-independent model, typically the output of the design phase, to the platform-specific model is not straightforward. The computational limitations of robotic platforms make existing model-to-text transformations of the AOSE methodologies obsolete and new transformations need to be defined. Towards this end, it is very important that the AOSE methodology delivers a platform-independent model with a clear and compact meta-model. Statecharts offer more compact semantics than PNPs [25], behavior trees [3], and hierarchical FSAs [15], and additionally capture both the functional and behavioral aspects of the system.

Finally, behavior specification is not a trivial task. The development of the simple player, which served as our running example, led to a statechart with 99 states in a hierarchy with a depth of 17 (Figure 7). This shows the added value of starting with the early ASEME models and particularly using the automatic transformation of liveness formulas to a statechart, as opposed to starting the design directly with a statechart CASE tool, such as StatEdit [12], or using a flat statechart model, such as the plan diagrams of MaSE [2].

[6]Boost is a set of free peer-reviewed portable C++ libraries: www.boost.org

[7]TUC RoboCup team Kouretes: www.kouretes.gr

[8]Foundation for Intelligent Physical Agents www.fipa.org

7. CONCLUSION

In this paper we showed how the ASEME model-driven AOSE methodology can be extended for multi-robot behavior specification. The modeler is assisted by the existing graphical and model transformation tools of ASEME and by the IAC2Monas transformation tool that allows the automated code generation for the defined behavior coupled with a generic multi-threaded statechart engine and a blackboard publish/subscribe messaging system.

We discussed the challenges we faced, to share our experience with any AOSE practitioner aiming to move to robotic agents' development. The solutions proposed in our work can serve as a first guide on how to go about addressing such challenges.

Our future work lies in enhancing the code generation tool with tight checking functionalities for minimizing user errors, e.g. for semantic validation of the transition expressions. Moreover, we plan to work on making the graphical editing tools more efficient and more flexible in visualizing and manipulating statecharts. The collection of our tools for ASEME-based robot software development, including the entire code of our RoboCup team, has been released to the community through www.kouretes.gr/aseme.

8. ACKNOWLEDGMENTS

The authors would like to thank Mr Antonis Argyriou, Mrs Aggeliki Topalidou-Kyniazopoulou, Mrs Shabana Shaikh, and all members of the Kouretes team for their valuable assistance. Also, Chipita S.A.–Molto for sponsoring our team.

9. REFERENCES

[1] D. Calisi, A. Censi, L. Iocchi, and D. Nardi. OpenRDK: A modular framework for robotic software development. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1872–1877, September 2008.

[2] S. DeLoach, E. T. Matson, and Y. Li. Applying agent oriented software engineering to cooperative robotics. In *Proceedings of the 15th International Florida Artificial Intelligence Research Society Conference (FLAIRS)*, pages 391–396. AAAI Press, May 2002.

[3] R. Dromey. From requirements to design: Formalizing the key steps. In *Proceedings of the First International Conference on Software Engineering and Formal Methods (SEFM)*, pages 2–11, September 2003.

[4] P. T. Eugster, P. A. Felber, R. Guerraoui, and A.-M. Kermarrec. The many faces of publish/subscribe. *ACM Computing Surveys*, 35:114–131, 2003.

[5] J. M. Gascuena and A. Fernandez-Caballero. Agent-oriented modeling and development of a person-following mobile robot. *Expert Systems with Applications*, 38(4):4280–4290, 2011.

[6] D. Harel and A. Naamad. The Statemate semantics of statecharts. *ACM Transactions on Software Engineering and Methodology*, 5:293–333, 1996.

[7] B. Hayes-Roth. A blackboard architecture for control. *Artificial Intelligence*, 26(3):251–321, 1985.

[8] J. Highsmith and M. Fowler. The agile manifesto. *Software Development Magazine*, 9(8):29–30, 2001.

[9] ISO/IEC. Extended Backus-Naur form (EBNF). 14977, 1996.

[10] H. Kitano, M. Asada, Y. Kuniyoshi, I. Noda, E. Osawa, and H. Matsubara. Robocup: A challenge problem for AI. *AI Magazine*, 18(1):73–85, 1997.

[11] A. G. Kleppe, J. Warmer, and W. Bast. *MDA Explained: The Model Driven Architecture: Practice and Promise*. Addison-Wesley, 2003.

[12] J. Murray. Specifying agent behaviors with UML statecharts and StatEdit. In *RoboCup 2003: Robot Soccer World Cup VII*, volume 3020 of *Lecture Notes in Computer Science*. Springer, 2004.

[13] O. Obst. Specifying rational agents with statecharts and utility functions. In *RoboCup 2001: Robot Soccer World Cup V*, volume 2377 of *Lecture Notes in Computer Science*. Springer, 2002.

[14] A. Paraschos. Monas: A flexible software architecture for robotic agents. Diploma thesis, Technical University of Crete, Greece, 2010.

[15] M. Risler. *Behavior Control for Single and Multiple Autonomous Agents Based on Hierarchical Finite State Machines*. PhD thesis, Technische Universität Darmstadt, Germany, 2009.

[16] T. Röfer et al. B-Human team report and code release, 2009. Only available online: www.b-human.de.

[17] C. Schlegel, T. Hassler, A. Lotz, and A. Steck. Robotic software systems: From code-driven to model-driven designs. In *Proceedings of the International Conference on Advanced Robotics (ICAR)*, pages 1–8, June 2009.

[18] N. Spanoudakis. *The Agent Systems Engineering Methodology (ASEME)*. PhD thesis, Paris Descartes University, France, 2009.

[19] N. I. Spanoudakis and P. Moraitis. The agent modeling language (AMOLA). In *Proceedings of the 13th International Conference on Artificial Intelligence: Methodology, Systems, and Applications (AIMSA)*, volume 5253 of *Lecture Notes in Computer Science*, pages 32–44. Springer, September 2008.

[20] N. I. Spanoudakis and P. Moraitis. Gaia agents implementation through models transformation. In *Proceedings of the 12th International Conference on Principles of Practice in Multi-Agent Systems (PRIMA)*, volume 5925 of *Lecture Notes in Computer Science*, pages 127–142. Springer, December 2009.

[21] N. I. Spanoudakis and P. Moraitis. Using ASEME methodology for model-driven agent systems development. In *Agent-Oriented Software Engineering XI, Revised Selected Papers of the 11th International Workshop AOSE 2010*, volume 6788 of *Lecture Notes in Computer Science*, pages 106–127. Springer, 2011.

[22] E. Vazaios. Narukom: A distributed, cross-platform, transparent communication framework for robotic teams. Diploma thesis, Technical University of Crete, Greece, 2010.

[23] M. Winikoff and L. Padgham. *Developing Intelligent Agent Systems: A Practical Guide*. Halsted Press, 2004.

[24] M. Wooldridge, N. R. Jennings, and D. Kinny. The Gaia methodology for agent-oriented analysis and design. *Autonomous Agents and Multi-Agent Systems*, 3(3):285–312, 2000.

[25] V. Ziparo, L. Iocchi, P. Lima, D. Nardi, and P. Palamara. Petri net plans. *Autonomous Agents and Multi-Agent Systems*, 23:344–383, 2011.

Session 5A
Robotics III

Active Visual Sensing and Collaboration on Mobile Robots using Hierarchical POMDPs

Shiqi Zhang
Department of Computer Science
Texas Tech University
s.zhang@ttu.edu

Mohan Sridharan
Department of Computer Science
Texas Tech University
mohan.sridharan@ttu.edu

ABSTRACT

A key challenge to widespread deployment of mobile robots in the real-world is the ability to robustly and autonomously sense the environment and collaborate with teammates. Real-world domains are characterized by partial observability, non-deterministic action outcomes and unforeseen changes, making autonomous sensing and collaboration a formidable challenge. This paper poses vision-based sensing, information processing and collaboration as an instance of probabilistic planning using partially observable Markov decision processes. Reliable, efficient and autonomous operation is achieved using a hierarchical decomposition that includes: (a) convolutional policies to exploit the local symmetry of high-level visual search; (b) adaptive observation functions, policy re-weighting, automatic belief propagation and online updates of the domain map for autonomous adaptation to domain changes; and (c) a probabilistic strategy for a team of robots to robustly share beliefs. All algorithms are evaluated in simulation and on physical robots localizing target objects in dynamic indoor domains.

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics

General Terms

Algorithms, Experimentation

Keywords

Integrated perception, cognition, and action; Robot planning (including action and motion planning); Robot teams, multi-robot systems, robot coordination.

1. INTRODUCTION

Autonomous and robust sensing and collaboration is a key challenge to widespread deployment of mobile robots in the real-world. Real-world application domains are characterized by partial observability, non-deterministic action outcomes and unforeseen dynamic changes. A robot equipped with multiple sensors (e.g., cameras and range finders) can use different algorithms to process sensory inputs with varying levels of reliability and computational complexity. It is not feasible for the robot to observe the entire domain or process all sensory inputs with all available algorithms and still respond to dynamic changes. At the same time, robust operation requires that the robot make best use of relevant information. Furthermore, each robot in a team can possess different capabilities and communication between robots can be unreliable. Autonomous and robust sensing and collaboration on robots deployed in the real-world is hence a formidable challenge.

This paper poses vision-based sensing and collaboration as a planning task and uses partially observable Markov decision processes (POMDPs) [8] to enable each robot in a team to tailor sensing and processing to the task at hand. Although POMDPs elegantly model the non-determinism and partial observability of real-world domains, the state space of these domains typically increases exponentially and even state of the art (approximate) POMDP solvers have high computational complexity [14, 19]. Our prior work introduced a hierarchical decomposition in POMDPs for reliable and efficient visual sensing and processing in simulation and simplistic tabletop scenarios [22, 23]. This paper builds on our prior work to enable a robot to autonomously direct sensing to relevant locations, and consider the reliability and complexity of available algorithms to determine the sequence of sensing and processing actions best suited to a given task. Each robot then shares beliefs (acquired by processing sensory cues) with teammates to collaborate robustly in real-world domains. The following novel contributions are made:

- Local symmetries in visual sensing are exploited to learn convolutional policies for efficient operation over large state spaces.
- Adaptive observation functions, policy re-weighting and automatic belief propagation in the hierarchy are used in conjunction with online revisions to domain map (based on range data) to enable the robot to adapt to dynamic changes.
- A probabilistic belief sharing strategy is used to enable a team of robots to merge individual and communicated beliefs to collaborate robustly despite unreliable communication.

These contributions enable the use of POMDPs for reliable, efficient and autonomous visual sensing and collaboration on mobile robots. All algorithms are evaluated in simulation and on physical robots deployed to localize target objects in dynamic indoor domains. The remainder of the paper is organized as follows. Section 2 summarizes related work, while Section 3 describes the hierarchical planning approach. Experimental results are described in Section 4, followed by conclusions in Section 5.

2. RELATED WORK

Research in vision, planning and robotics has produced sophisticated algorithms for planning a pipeline of visual operators for a high-level goal. Many such algorithms use deterministic action models whose preconditions and effects are propositions that need to be true apriori, or are made true by executing the operator. However, such formulations are insufficient for application domains with partially observable state and non-deterministic action outcomes.

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

In vision research, image interpretation has been modeled using MDPs and POMDPs. Li et al. [12] used human-annotated images to determine the reward structure, explore the state space and compute value functions—actions that maximize the learned functions are chosen during online operation. Similarly, active sensing has been used to decide sensor placement and information processing, using particle filters and relative entropy maximization for estimating a joint multitarget probability density [10]. Sensor placements in spatial phenomena have also been modeled as Gaussian processes using submodular functions [9]. However, many visual planning tasks are not submodular, and it is difficult to model probability densities using manual feedback over many trials on robots.

Since a POMDP formulation can become intractable due to the exponential state explosion of real-world domains, researchers have focused on imposing structure on application domains. Pineau and Thrun [16] proposed a hierarchical approach for behavior control of a robot assistant. The top level action is a collection of simpler actions modeled as smaller POMDPs and solved completely to enable bottom-up planning and top-down plan execution. Similar approaches have been used for robot navigation [7] but a significant amount of data for the hierarchy and model creation is hand-coded. Recent work has focused on learning POMDP observation models [1]; using information maximization for POMDP-based visual search [4, 23], and developing factored representations and faster POMDP solvers [14, 19]. Researchers have also focused on integrating human input in POMDPs for human-robot interaction [18]. However, these methods are still not suitable for dynamic domains with large state spaces, and do not enable automatic model creation and belief propagation that is essential for robot domains.

Many algorithms continue to be developed for multiagent and multirobot collaboration in a variety of domains [15]. Sophisticated algorithms have also been developed recently for using decentralized POMDPs (Dec-POMDPs) for multiagent and multirobot collaboration [11]. However, the computational complexity of these formulations is more than that of POMDP formulations [2]. Research has also shown that using complex communication strategies does not necessarily improve task completion times [21]. This paper addresses these challenges using hierarchical POMDPs that enable autonomous active visual sensing on each robot and robust collaboration between a team of robots.

3. PROBLEM FORMULATION

Figure 1 summarizes the POMDP hierarchy for visual sensing, processing and collaboration. Each robot uses the hierarchy to locate one or more target objects. The top-level visual sensing (**VS**)-**POMDP** determines the sequence of 3D scenes to process to locate a specific target, as described in Sections 3.1–3.3. For each chosen scene, the scene processing (**SP**)-**POMDP** determines the sequence of regions to process in a sequence of images using the appropriate set of algorithms. The SP-POMDP has one or two layers depending on the characterization of the learned object models, as described in Section 3.4. The hierarchy is then augmented with a *communication layer* that enables each robot in a team to share beliefs with teammates to collaborate robustly despite unreliable communication, as described in Section 3.5.

3.1 POMDP Planning

In real-world domains, the robot has to move and analyze different scenes to locate target objects that can exist in different locations. Consider the situation where a robot has learned a domain map [6] and has to locate a specific target. The 3D area is represented as a discrete 2D *occupancy grid* and each grid cell stores the probability of occurrence of the target object. The VS-POMDP

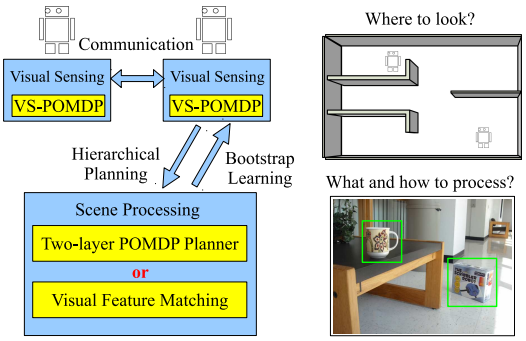


Figure 1: Overview of POMDP hierarchy for target localization.

poses sensing as the task of maximizing information gain, i.e., reducing the belief state entropy in a grid with N cells. The POMDP tuple $\langle S, A, T, Z, O, R\rangle$ is defined as:

- $S : s_i,\ i \in [1, N]$ is the state vector; s_i corresponds to the event that the target is in grid cell i.
- $A : a_i, i \in [1, N]$ is the set of actions. Executing a_i causes the robot to move to and analyze grid cell i.
- $T : S \times A \times S \rightarrow [0, 1]$ is the state transition function. It is an identity matrix here because actions do not change state.
- Z : {present, absent} is the observation set that indicates if the target is detected.
- $O : S \times A \times Z \rightarrow [0, 1]$ is the observation function (see below).
- $R : S \times A \rightarrow \mathbb{R}$ is the reward specification that is based on belief entropy (see below).

The robot maintains a *belief state*, a probability distribution over the state. The *entropy* of belief distribution B_t is given by:

$$\mathscr{H}(B_t) = -\sum_{i=1}^{N} b_t^i log(b_t^i) \tag{1}$$

where b^i is the i^{th} entry of the belief distributed over the N grid cells. With no prior knowledge of target location, the belief is uniformly distributed and entropy is maximum. The VS-POMDP aims to choose actions that significantly reduce the entropy by causing the belief distribution to converge to likely target locations. The reward of action a_t at time t is hence defined as the entropy reduction between belief state B_{t-1} and the resultant belief state B_t:

$$\begin{aligned} R(a_t) :=& \mathscr{H}(B_{t-1}) - \mathscr{H}(B_t) \\ =& \sum_k b_t^k log(b_t^k) - \sum_j b_{t-1}^j log(b_{t-1}^j) \end{aligned} \tag{2}$$

The observation function models the probability of target detection as a function of the robot position and target position:

$$\begin{aligned} &\textbf{if } isBlocked(s_j, a_k) \\ &\quad O(z_i = present, s_j, a_k) = Pr(z_i = present | s_j, a_k) = \beta \\ &\textbf{else} \\ &\quad O(z_i = present, s_j, a_k) = \eta \cdot exp\{-\lambda \mu^2 / 2\sigma^2\} \\ &O(z_i = absent, s_j, a_k) = 1 - O(z_i = present, s_j, a_k) \end{aligned} \tag{3}$$

where the probability of observation "present" in cell i given that the target is in cell j and the focus is on cell k, i.e., $p(z_i|s_j, a_k)$, is a Gaussian distribution whose mean depends on the target location, the grid cell being examined and the field of view: $\mu = f_\mu(s_j, a_k)$. The variance of the Gaussian represents the sensitivity of sensory cues to the object's distance from the sensor—there is more uncertainty associated with the observation of a target at a greater distance. The factor η is a normalizer. If there is any obstacle be-

tween the robot and the target, i.e., $isBlocked(s_j, a_k)$, β is a small probability that the target can still be observed. This observation function is used to perform belief updates after sensing actions provide observations, and to generate observations in the simulated experiments. Given these model parameters, belief update in the VS-POMDP proceeds as follows:

$$B_{t+1}(s') = \frac{O(s', a_{t+1}, o_{t+1}) \sum_s T(s, a_{t+1}, s') \cdot B_t(s)}{p(o_{t+1}|a_{t+1}, b_t)} \quad (4)$$

POMDP solvers take such a model and compute a *policy* that maps belief states to actions: $\pi : B_t \mapsto a_{t+1}$. In the VS-POMDP, the computed policy has to minimize entropy in B_t over a planning horizon. Policy gradient algorithms are used to compute the policy in the form of stochastic action choices, i.e., the policy is learned as a matrix of "weights" that are used (during plan execution) to probabilistically choose an action for specific belief states [3]. Actions in the VS-POMDP require the robot to physically move between grid cells, expending time and effort. Instead of the formulation described above, motion costs are addressed in a post-processing step, as described in Section 3.3.

3.2 Convolutional Policy

In real-world domains, the state space of the VS-POMDP can increase exponentially, making it intractable to compute the policy in real-time even with sophisticated solvers. This challenge is addressed by exploiting the local shift and rotation symmetries of visual processing. Specifically, if the robot is analyzing a specific grid cell, only the beliefs immediately around that grid cell change substantially, i.e., the performance is a function of (and can affect) only a small number of surrounding cells. The robot captures this local influence by learning a *policy kernel* based on a *baseline* policy for a map with a small number of grid cells. The policy for a larger map with a larger number of grid cells is generated automatically by an inexpensive convolution operation. This section describes the creation of policy kernels and the use of convolutional policies for efficient sensing and processing.

3.2.1 Kernel Extraction

Consider the stochastic baseline policy generated for a 5×5 map, which has 25 states and 25 actions. In the 2D matrix of action weights, each column corresponds to an action and each row corresponds to a state. The matrix is re-organized into layers, where each layer corresponds to action weights for a particular state and is a represented as a 2D matrix of the same size as the map. This re-organization enables the robot to use the local symmetries (i.e., shift and rotation invariance) to extract a kernel without significant loss of information:

$$\bar{K}(s) = (\pi^V \otimes C_m^K)(s) = \int \pi^V(\tilde{s}) C_m^K(s - \tilde{s}) d\tilde{s}, \quad (5)$$
$$K = (\sum_{states} \bar{K}) \cdot / W$$

where $\bar{K}$ is the un-normalized kernel, π^V is baseline policy generated for the VS-POMDP over the 5×5 map and C_m^K is the convolution mask of the same size as the target kernel. Since the mask only considers action weights within a local region, the layers of the resultant kernel are summed up and normalized using W, a matrix that stores the count of the number of accumulated weights across all layers. For instance, a 3×3 policy kernel is computed by convolving a 3×3 mask with the 5×5 policy layers and normalizing the weights in the region covered by the mask.

The computed kernel does not assign action weights to grid cells further away from the center of the convolution mask. Since these action weights are usually much lower than values in the kernel, they can all be set to a small default value:

$$w^d = \frac{\sum_{actions} \sum_{states} \pi^V - \sum_{states} \bar{K}}{N_{actions} \times N_{states} - \sum W} \quad (6)$$

where the default action weight w^d is a function of the number of states (N_{states}) and actions ($N_{actions}$). To prevent the summation of "small weights" from overwhelming the kernel's weights when generating policies for large maps, w^d is revised to make the ratio of importance assigned to the area covered and left uncovered by the kernel to be similar over maps of different sizes:

$$\hat{w}^d = w^d - ln(\frac{N_{states}^E - sz(W)}{N_{states}^K - sz(W)}) \quad (7)$$

where N_{states}^E and N_{states}^K are the number of states in the large map and kernel respectively, and $sz(W)$ is the number of entries in W.

3.2.2 Policy Extension

Once a policy kernel has been learned, it can be used to efficiently compute the convolutional policy for a larger map:

$$\pi_C^V(s) = (K \otimes C_m^E)(s) = \int K(\tilde{s}) C_m^E(s - \tilde{s}) d\tilde{s} \quad (8)$$

where π_C^V is the convolutional policy, K is the policy kernel and C_m^E is the convolution mask of the same size as the target map. For instance, for a 10×10 map, C_m^E is a 10×10 mask over which the 3×3 policy kernel is convolved. The desired policy is generated one layer at a time by centering the kernel on the state represented by the layer. Since the kernel covers only grid-cells in a small area, other cells are assigned the weight computed in Equation 7 and the resultant policy is normalized. Although it may take some time for the robot to learn a baseline policy for a small map, it is a one-time computation. The kernel extracted from a baseline policy needs to be revised only when the robot's sensors change substantially.

3.3 Motion Costs and Path Planning

Unlike visual search over an image, a mobile robot has to physically move between grid cells. The movement takes time and is associated with unreliability that has a cumulative effect as the distance traveled increases. Each action is hence assigned a cost proportional to the distance to be traveled by revising the action's policy weights during policy execution:

$$\hat{w}(i) = w(i) \frac{1}{1 + \frac{d_{A^*}(a_i, a_j)}{\text{speed}}} \quad (9)$$

where $d_{A^*}(a_i, a_j)$ is the distance between the current grid cell and the candidate grid cell, which is computed using the A^* search algorithm [20]. The A^* search includes a heuristic cost to the target grid cell and a path cost to account for obstacles (e.g., walls) in the domain map. The revised policy trades off the expected likelihood of locating the target in a specific grid cell against the cost of traveling to that location. When the domain map changes due to changes in object configurations (e.g., objects are moved and/or new obstacles are created), the robot automatically revises the map using laser-based simultaneous localization and mapping (SLAM) algorithms. The modified map is used to recompute distances between grid cells and revise action weights for subsequent computations.

In addition to revising action weights to model motion-based costs, hill-climbing is used to make the search more efficient in large maps. Consider Figure 2, which shows a domain map (similar to Figure 8) discretized into grid cells. The green grid is the current position of the robot after executing the most recent action. At this point, there are three grid cells in the map with significantly

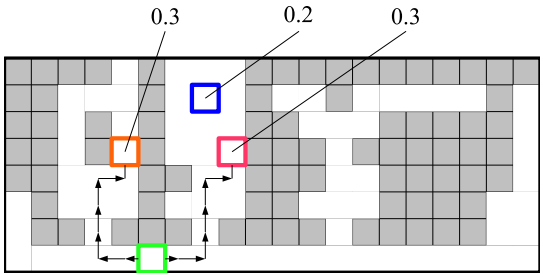


Figure 2: Illustration of hot-spot detection.

higher weights than the other cells: the orange and pink grids have $w = 0.3$ and the blue grid has $w = 0.2$. Since the robot's current position is equidistant from the pink and orange grids, these grids have an equal chance of being the next grid cell visited by the robot. However, given that the robot has three valid candidates of similar relevance, it makes sense to visit the pink cell first because it is also close to the blue grid cell. Instead of looking for a grid cell with the largest $\hat{w}$ (Equation 9), the robot therefore selects the path through the candidate grid cells that has the largest summation of $\hat{w}$ values. Since it is computationally expensive to estimate an optimal path by evaluating all paths through all grid cells in a large map, the robot detects "hot-spots", i.e., grid cells with sufficiently large beliefs, and plans a path through them.

To compute hot-spots, N seeds are randomly selected and then refined based on hill-climbing to arrive at local maxima, i.e., cells similar to the orange, blue and pink grids in Figure 2. These hot-spots are considered to be the interesting areas for further analysis. The robot then computes the values of paths w^p through combinations of these hot-spots:

$$w^p([h_0, h_1, \ldots, h_N]) = \sum_{i=1}^{N} f(w_i, \sum_{j=1}^{i} d_{A^*}(h_{j-1}, h_j)) \qquad (10)$$

where, h_n is the nth hot-spot, h_0 is the current position of the robot and other entries are chosen by hill-climbing. The function f is defined in Equation 9. In Figure 2, the values of the *pink-blue-orange* and *orange-pink-blue* paths are 0.0672 and 0.0591 respectively, making the pink grid cell the most likely choice for being analyzed next. This path planning does *not* imply that the robot will move through all the hot-spots—once a robot arrives at a grid cell, the corresponding observation revises the belief distribution and hence the planned path. The path planning ensures that the robot's attention is directed towards the most interesting grid cells.

3.4 Scene Processing

Invoking the VS-POMDP policy computed for a specific target causes a 3D scene to be chosen for analysis. The robot moves and captures images of this scene. As stated earlier, there are two options for scene processing depending on the scene complexity and learned object models—specific examples are provided in Section 4. In uncluttered scenes with unique objects, the SP-POMDP is a two layered POMDP as described in [22]. Each input image is analyzed to extract salient regions of interest (ROI). Each ROI is modeled as a lower-level (**LL**)-**POMDP**, where actions are information processing operators (e.g., to detect color or shape). The LL policy provides the best sequence of operators to apply on a specific ROI to detect the target. The LL policies of all image ROIs are used to automatically create a high-level (**HL**)-**POMDP**. Executing an action in the corresponding HL policy directs robot's attention to a specific ROI. The result of executing the corresponding LL policy causes an HL belief update and action choice until presence or absence of the target in the image is determined. In cluttered scenes with sophisticated learned object models, the robot may need to process the entire image. Scene processing is then reduced to a single POMDP over the image. With either version of SP-POMDP, the result of scene processing causes a belief update in the VS-POMDP and subsequent analysis of grid cells until the target is found or a time limit is exceeded. The entire hierarchy operates automatically and efficiently for dynamic domains.

3.5 Multirobot Collaboration

Consider (next) a team of X robots trying to locate Y targets. Each robot maintains a belief vector for each target, and uses the hierarchical POMDPs to detect each target. This section describes an algorithm for a team of robots to share beliefs and collaborate to locate all targets reliably and efficiently.

We assume that the targets are visually distinguishable and that the observations of different targets are independent of each other. Each robot now stores a data structure:

$$\{B_i, f_i\}, \forall i \in [1, |TL|] \qquad (11)$$

where B_i is the belief vector for a specific target i among the list of target objects (TL) and f_i is a binary flag that indicates discovery of a target. The robot also stores an action map $\mathscr{M}$, a vector of the same size as the belief vector. Each entry in this vector stores the number of times the robot has visited the corresponding grid cell:

$$\mathscr{M} = \langle m_1, \cdots, m_N \rangle \qquad (12)$$

where m_i is the count of the number of times grid-cell i has been visited. For moving targets, values in the action map decay over time if they are not reinforced by more recent visits. Each robot uses the POMDP hierarchy to update the appropriate belief vectors based on observations. After the belief update, each robot shares the belief information with its teammates by broadcasting a package that includes its current belief vectors ($\forall i\, B_i$), discovery flags ($\forall i\, f_i$) and the action map ($\mathscr{M}$).

There is uncertainty associated with sensing on each robot and communication between robots—the information from a teammate (when received successfully) may reinforce or contradict the information acquired by the robot by processing sensory inputs. At the same time, the communicated estimates provide useful information about map locations that the robot has not visited. Each robot hence merges own and communicated beliefs by assigning a trust factor to beliefs based on whether the robot that generated this belief vector has recently observed the corresponding map region:

$$b_i^{j,own} = \frac{m_i^{j,own} \cdot b_i^{j,own} + m_i^{j,comm} \cdot b_i^{j,comm}}{m_i^{j,own} + m_i^{j,comm}} \qquad (13)$$
$$\forall j \in [1, N], \quad \forall i \in [1, |TL|]$$

where b_i^j is j^{th} entry of the belief vector of the i^{th} target, while $m_i^{j,own}$ and $m_i^{j,comm}$ are action map entries of the robot and the teammate whose communicated belief is being merged. Although this merging process can be sensitive to processing order, it works well in practice. Next, the target discovery flags are updated:

$$\mathscr{F} = \{ f_i^{own} || f_i^{comm}; \forall i \in [1, |TL|] \} \qquad (14)$$

where each target is considered to be found when at least one robot has localized it. Once a target is discovered, a robot that requires a new target chooses an undiscovered object from the list ($|TL|$):

$$targetID = argmax_i \{ \max_j B_i(j) \} \qquad (15)$$

where the robot chooses the target object whose location it is most certain about, i.e., the target that is likely to require the least amount

of work to localize. The robot makes this choice based on current beliefs that include the beliefs communicated by teammates. This target selection approach (intentionally) includes some overlap of targets among robots to account for unreliable communication, but robots in the team distribute tasks and rarely go to the same target. Furthermore, an additional cost is included to trade-off distance of travel against expected likelihood of locating the target (or priority of target, if known), similar to Equation 9.

4. EXPERIMENTAL RESULTS

This section describes the results of experiments performed to evaluate the robot's ability to: (a) use *convolutional policies* and the POMDP hierarchy for reliable, autonomous and efficient visual sensing and processing in complex domains; and (b) probabilistically merge own beliefs with communicated beliefs of teammates to achieve robust collaboration. Experiments were hence conducted in simulation and on robots to evaluate the following hypotheses: (I) the constrained convolution (CC) policy is more efficient than the non-convolutional (i.e., baseline) policy while providing similar accuracy; (II) the POMDP hierarchy results in better target localization in comparison to heuristic search strategies; and (III) the belief merging strategy enables a team of robots to share beliefs and collaborate robustly despite unreliable communication.

4.1 Experimental Setup

Before describing the experimental results, this section describes the initial setup and the modifications necessary for experimental trials on robots. The initial setup consisted of a semi-supervised learning phase, where some objects with known labels were placed in front of the robot. The robot applied different processing operators on images of these objects to learn object models and some model parameters of the VS-POMDP and SP-POMDP (e.g., observation functions, reward specifications). Examples of learned object models are described in Sections 4.3.1 and 4.3.2. The robot also used data from a laser range finder to learn a domain map that was revised continuously during experimental trials.

For any detected object, the robot computes the relative distance and bearing using geometric transforms. However, including orientation as a parameter in the observation set will destroy the local symmetry in visual sensing. The belief update in Equation 4 was therefore modified as:

if $\neg target$ (16)

$$B(s') = \frac{O(s',a,o)\sum_{s\in S} T(s,a,s')b(s)}{Pr(o|a,b)} = \frac{O(s',a,o)b(s)}{Pr(o|a,b)}$$

else

$$B(s') = \frac{O(s',\hat{a},o)\sum_{s\in S} T(s,\hat{a},s')b(s)}{Pr(o|\hat{a},b)} = \frac{O(s',\hat{a},o)b(s)}{Pr(o|\hat{a},b)}$$

where $B(s')$ is the updated belief for state s' after action a. Since the transition functions are identity matrices, the update equation can be simplified as shown. The robot's estimate of its own position and the relative distance and bearing of a detected target are used to find the target's global location in the domain map. The belief is then updated as if the action corresponding to this global location had been executed: $\hat{a}$. This belief update scheme also models the fact that false positives are rare while false negatives are common when sensing (or processing) actions are executed on mobile robots. Furthermore, a robot moving between grid cells may receive sensory inputs relevant to the current task, e.g., it may unexpectedly have the target in its field of view. The robot therefore periodically processes input images at low-resolution to update the current belief.

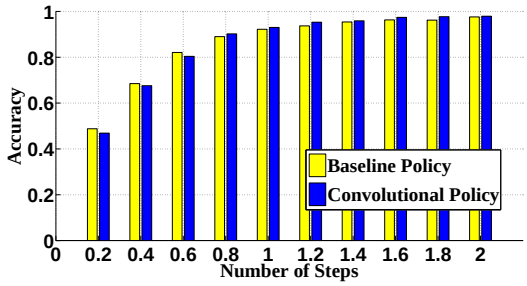


Figure 3: CC policy performs as good as the baseline policy.

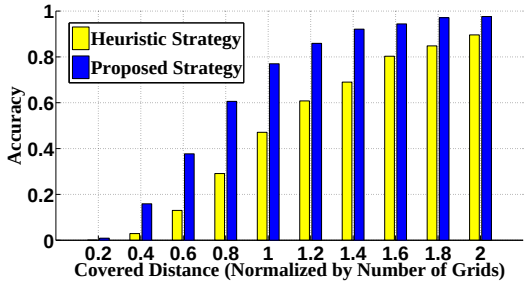


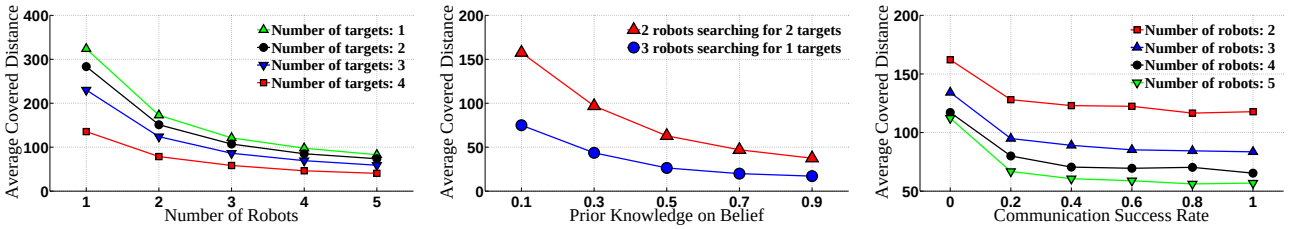
Figure 4: CC policy performs better than a heuristic strategy.

4.2 Simulation Experiments

All three hypotheses were evaluated extensively in simulation using domain maps that represented different sections of the map shown in Figure 8. Each data point in the figures in this section is the average of 1000 simulated trials. To evaluate hypothesis I, a baseline policy computed for a 5×5 map was used to extract a policy kernel that was used to compute policies for larger maps. Figure 3 compares the CC policy against the baseline policy for a 7×7 map—the x-axis shows the number of times the policy was invoked, as a fraction of the number of states. In each trial, the initial positions of the target and the robot were set randomly and the trial was deemed successful if the target was localized correctly. There is no statistically significant difference in the target localization accuracies of the CC and baseline policies. However, it takes a few hours to compute the baseline policy for the 7×7 map.

Hypothesis II was evaluated by comparing the CC policy's performance against a heuristic policy that makes greedy action choices or selects random actions based on the presence/absence of prior knowledge. The results shown in Figure 4 correspond to a 15×15 convolutional policy generated from a 5×5 kernel. The locations of the robot and the target were randomly selected for each trial. Existence of prior knowledge was simulated by adding bias to the initial belief—70% of the belief was uniformly distributed over all grid cells, while the remaining 30% was Gaussian-distributed around the target. To generate the data points in Figure 4, trials were terminated after a certain distance had been traveled and the grid cell with the largest belief value was taken to be the target's location. The robot's performance is scored as the weighted distance between the actual and detected locations of the target. Figure 4 shows that the CC policy significantly reduces the number of action steps required to locate the target with high accuracy.

Experiments were conducted next to evaluate the multirobot collaboration capability, i.e., hypothesis III. Assuming that all robots in a team move at the same speed, the average distance moved by the robots in a team (in an episode/trial) was used as a measure of the team's performance. In each trial, robots and targets were placed randomly in a grid map, with no more than one robot or target in each grid-cell. A Gaussian bias (20%) was added to the initial



(a) Multirobot search. (b) Effect of Bias. (c) Effect of CSR.

Figure 5: (a) Belief merging and hierarchical POMDPs result in robust multirobot collaboration; (b) Performance improves if prior information is incorporated; and (c)Performance is robust to dropped communication packages.

Table 1: Proposed algorithms enable a robot team to localize targets more accurately than random and heuristic search strategies.

Algorithm	Normalized covered distance			
	0.5	1.0	1.5	2.0
Random	0.033	0.171	0.382	0.537
Heuristic	0.079	0.334	0.549	0.817
Proposed	0.153	0.544	0.825	0.957

belief in a 3×3 area around every target—the belief vector was then normalized. When the belief in a grid cell exceeded 0.9, the grid cell was assumed to contain a target. To simulate unreliable communication, a *communication success rate* (CSR) parameter was introduced and set to 0.5, i.e., every other broadcasted package was not received. Figure 5(a) shows results for different combinations of robots and targets in a 15×15 grid map—the robots collaborate effectively to find the targets. Similar results were obtained for grid maps of different sizes (4×4 to 25×25) that represent different sections of the real-world office domain shown in Figure 8.

Next, the ability of the proposed collaboration algorithm to incorporate prior knowledge of target locations was evaluated. Figure 5(b) shows examples of the team's performance for a specific number of robots and targets as a function of the bias in the initial belief. As expected, the performance improves, i.e., the robots are able to localize targets faster, as more information about the locations of targets is made available.

Next, the effect of communication uncertainty on multirobot collaboration was measured. Figure 5(c) shows results of experiments as a function of varying CSR, where robot teams were asked to locate two targets. Though a low likelihood of successful communication hurts the team's performance, the target localization capability soon stabilizes and is then no longer sensitive to the CSR.

Table 1 shows results of an experiment where two robots localized two targets in a 15×15 map. The initial positions of robots and targets were randomly assigned in each trial. The POMDP-based approach is compared to a policy that randomly selects actions and assigns targets to robots, and a heuristic policy which selects targets and actions based on the grid cell with the largest belief. To simulate more realistic scenarios, prior belief was assigned to multiple areas in the map (including the target location). The proposed approach results in significantly better performance, with the robots traveling a much smaller distance to localize targets with high accuracy. Over extensive simulation experiments (and robot trials, see below) in different maps (3×3 to 25×25), using the hierarchical POMDP and collaboration strategy enables a team of robots to collaborate and localize target objects reliably and efficiently.

Figure 6 is a pictorial representation of the proposed approach for multirobot collaboration, with two robots repeatedly localizing two targets in a 20×20 map with obstacles. The robots had no prior knowledge of target locations. Intuitively, each robot should first look around its starting position and then explore other areas. Once

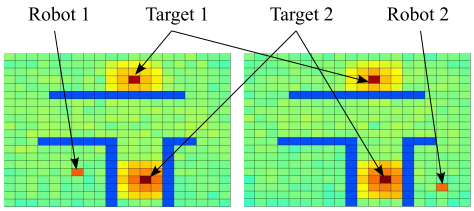
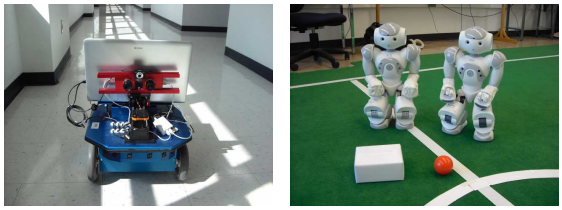


Figure 6: Simulated trials with 2 robots and 2 targets. Obstacles are shown in *blue*, targets in *dark red* and robot starting positions in *red*. Other cells show the number of times they were visited using colors ranging from *blue* to *red* along the visible spectrum.

a target is sighted, the robot should localize the target accurately. The actions taken by the robots are recorded over 100 simulated trials—each trial ends when the targets are located. In Figure 6, each grid cell's color changes from *blue* to *red* along the visible spectrum based on the relative number of visits by a robot—results are shown separately for each robot. Figure 6 shows that obstacles are avoided and grid cells near the targets are visited more often than other grid cells. In the absence of prior knowledge, there is no clear path from initial robot positions to the targets. The radius of the yellow region reflects the largest distance of effective observation. The two robots start searching for different targets in different trials, but there are hardly any trials when they both go for the same target. Similar performance is observed for different grid maps with different numbers of targets and robots.

4.3 Robot Experiments



(a) Erratic robot (b) Nao robots

Figure 7: Robot platforms used in experiments.

Experiments were conducted on a wheeled robot and a team of humanoid robots to test the proposed algorithms for reliable, efficient and autonomous sensing and collaboration.

4.3.1 *Experiments on Wheeled Robot*

The algorithms for POMDP-based visual sensing and processing were evaluated on the *Erratic* robot platform shown in Figure 7(a). This robot is equipped with stereo and monocular cameras, in addition to a laser range finder that can provide range information over an angular range of $\pm 135^o$ for a distance of $30m$.

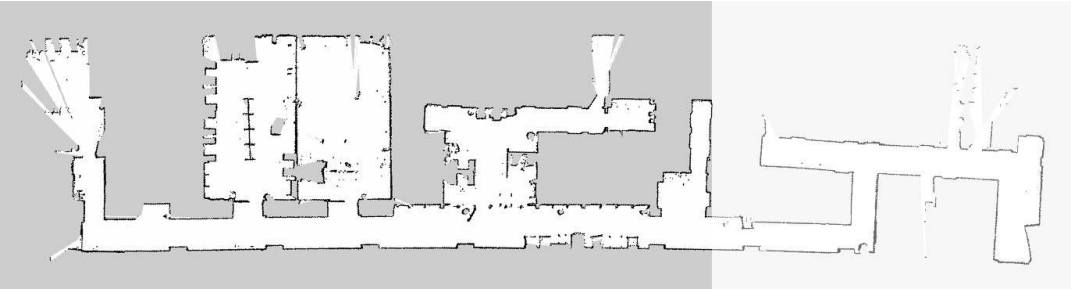


Figure 8: Occupancy-grid map of the third floor of the Computer Science department at Texas Tech University.

All processing is performed using an on-board dual-core 2.6*GHz* processor. The robot was used to conduct experiments in an indoor office domain—the corresponding occupancy-grid map was generated using a SLAM algorithm, as shown in Figure 8. This map corresponds to an entire floor of the CS department at Texas Tech University—it has three research labs, 13 faculty offices and a conference room. The experiments reported below were mostly conducted over the shaded portion of this map, which includes all research labs and nine rooms—this region was discretized into cells to form the grid map.

Given the complexity of the domain, objects were characterized using color distributions and the Binary Robust Independent Elementary Features (BRIEF) [5], i.e., local image gradients. Although BRIEF features are not inherently rotation and scale invariant, images of an object (captured during the learning phase) are automatically rotated and scaled to generate a set of images that encapsulate a range of rotations and scale changes—features extracted from these images are used to populate the object model. Figure 9 shows a screenshot of local feature detection and matching on a test object. Target objects consist of *boxes*, *cups*, *books* and other *robots* in complex (i.e., cluttered) backgrounds.

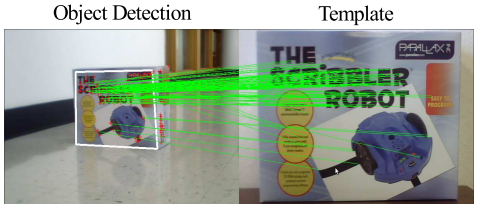


Figure 9: BRIEF descriptor.

To enable modular software architecture, the popular Robotics Operating System (ROS) [17] was installed on the robot and the algorithms described above were implemented on top of ROS. Figure 10 presents an overview of the implementation—it is a subset of the graph generated by the ROS command `<rxgraph>`. The planning algorithms are placed within the *vs_planner* node that is the control center of the system. It repeatedly accepts messages from the *vs_vision* node, which processes input images to provide the ID of any detected object, in addition to relative distance, bearing and detection probability, in the `<v_pack>` package. Belief updates occur under two situations: (1) robot arrives at a desired grid cell and processes some images of the scene—updates consider presence or absence of the target object; or (2) robot detects the target by processing images during navigation to a desired grid cell. The planner node sends the coordinates of any desired grid cell to the motion control node *move_base* and then waits for a response from the node, which can be one of: `arrived`, `canceled` or `not-arrived`. The `not-arrived` response is usually caused by a dynamic change in the environment, e.g., a door being closed, which makes an office unavailable to the robot. The node of the platform driver *erratic_base_driver* moves the robot platform based on the velocity command `cmd_vel`. The *hokuyo_node* provides the laser (range) readings to the motion control node and the localization node *amcl*. The *amcl* node computes the robot's `pos` (position and orientation) and the *map_server* revises the domain `map` continuously.

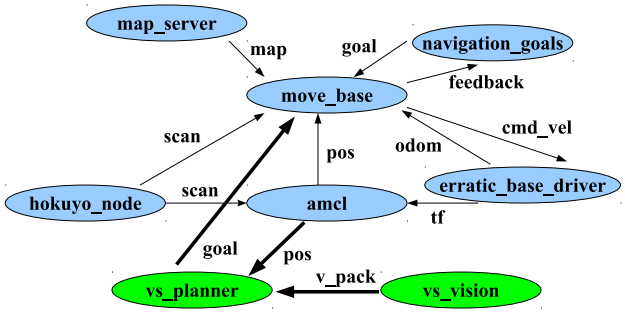


Figure 10: Node connections in ROS.

Over a sequence of 40 trials, the robot successfully identified the desired target objects. The robot only fails when a valid path to the target does not exist. The performance was significantly better than the heuristic search strategy used in Table 1. Videos of the robot's performance can be viewed online: `www.cs.ttu.edu/~smohan/Movies/Planning/visplan_aamas12.mp4`

4.3.2 *Experiments on Humanoid Robots*

The humanoid Nao robots [13] were used for multirobot collaboration experiments because multiple wheeled robots were not available. Since stable navigation is a challenge on humanoids, experiments were conducted in the robot soccer domain, where a team of robots play a competitive game of soccer on a $4m \times 6m$ indoor soccer field. This moderately constrained domain still captures all the collaboration challenges we seek to address. Each robot has a domain map and localizes based on domain landmarks such as goals and field corners (whose positions in the map are known) detected in input images. All computation is performed on-board the robots using a 500*MHz* processor.

Target objects include boxes and balls of different colors and shapes, as shown in Figure 7(b). Since objects are composed of homogeneous colors, gradient features cannot be used to learn object models. The robot has to process 30 frames/sec and computational resources are limited. Algorithms that detect object color and shape were hence used. Scene processing was modeled as a two-layered POMDP, with a POMDP that selects operators to apply on each salient region of interest in an image, and a POMDP that controls the selection of image ROIs for processing. The transfer of con-

trol between SP-POMDP and VS-POMDP occurred as described in Section 3.4. Obstacles were artificially introduced to force the robot to walk around to see the desired targets.

Experiments consisted of 25 trials, where a team of robots had to detect and localize one or more targets. The robots successfully localized all targets in all trials, and the performance was significantly better than the heuristic (i.e. greedy) policy for target and action selection, similar to the results reported in Table 1. The collaboration strategy was also robust to sudden changes in the team composition. For instance, when a robot was suddenly introduced in an existing team of robots, the new robot automatically (and quickly) chose to search for a relevant target using the communicated beliefs of teammates. Similarly, when a robot was removed from the team, the remaining robots automatically distributed the targets among themselves. These experiments show that the robots are able to use visual cues to reliably, efficiently and autonomously sense the environment and collaborate with teammates.

5. CONCLUSION

This paper described an approach for reliable, efficient and autonomous visual sensing and multirobot collaboration. A hierarchical POMDP with convolutional policies, adaptive observation functions, policy re-weighting and automatic belief propagation enables each robot to adapt sensing and information processing to that task at hand in dynamically changing environments. Each robot shares its beliefs with teammates and the multirobot collaboration algorithm enables the robot to merge its beliefs with the communicated beliefs of teammates. As a result, a team of mobile robots is able to collaborate robustly in simulation and in the real-world. The experiments reported in this paper assumed that robots have similar actuation capabilities. One direction of further investigation is to model and incorporate the sensing and actuation capabilities of heterogeneous robot platforms in the collaboration algorithm. Experiments will also be conducted using a larger number of physical robots and targets. Furthermore, the proposed hierarchy will be adapted to inputs from other sensors on mobile robot platforms. The ultimate goal is to enable reliable, efficient and autonomous multirobot (and human-robot) interaction in complex and dynamic real-world application domains.

Acknowledgment

This work was supported in part by the ONR Science of Autonomy award N00014-09-1-0658.

6. REFERENCES

[1] A. Atrash and J. Pineau. A Bayesian Method for Learning POMDP Observation Parameters for Robot Interaction Management Systems. In *The POMDP Practitioners Workshop*, 2010.

[2] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The Complexity of Decentralized Control of Markov Decision Processes. *Mathematics of Operations Research*, 27(4), November 2002.

[3] O. Buffet and D. Aberdeen. The Factored Policy-Gradient Planner. *Artificial Intelligence*, 173(5-6):722–747, 2009.

[4] N. J. Butko and J. R. Movellan. I-POMDP: An Infomax Model of Eye Movement. In *The IEEE International Conference on Development and Learning (ICDL)*, 2008.

[5] M. Calonder, V. Lepetit, C. Strecha, and P. Fua. BRIEF: Binary Robust Independent Elementary Features. In *European Conference on Computer Vision*, September 2010.

[6] G. Dissanayake, P. Newman, and S. Clark. A Solution to the Simultaneous Localization and Map Building (SLAM) Problem. *Transactions on Robotics and Automation*, 17(3):229–241, 2001.

[7] A. F. Foka and P. E. Trahanias. Real-time Hierarchical POMDPs for Autonomous Robot Navigation. In *IJCAI Workshop on Reasoning with Uncertainty in Robotics*, 2005.

[8] L. Kaelbling, M. Littman, and A. Cassandra. Planning and Acting in Partially Observable Stochastic Domains. *Artificial Intelligence*, 101:99–134, 1998.

[9] A. Krause, A. Singh, and C. Guestrin. Near-optimal Sensor Placements in Gaussian Processes: Theory, Efficient Algorithms and Empirical Studies. *JMLR*, 9:235–284, 2008.

[10] C. Kreucher, K. Kastella, and A. Hero. Sensor Management using An Active Sensing Approach. *IEEE Transactions on Signal Processing*, 85(3):607–624, 2005.

[11] J. Kwak, R. Yang, Z. Yin, M. Taylor, and M. Tambe. Teamwork and Coordination under Model Uncertainty in DEC-POMDPs. In *The AAAI Workshop on Interactive Decision Theory and Game Theory*, 2010.

[12] L. Li, V. Bulitko, R. Greiner, and I. Levner. Improving an Adaptive Image Interpretation System by Leveraging. In *Australian and New Zealand Conference on Intelligent Information Systems*, 2003.

[13] Nao. The Aldebaran Nao Robots, 2008. `http://www.aldebaran-robotics.com/`.

[14] S. C. Ong, S. W. Png, D. Hsu, and W. S. Lee. Planning Under Uncertainty for Robotic Tasks with Mixed Observability. *International Journal of Robotics Research*, 29(8):1053–1068, July 2010.

[15] L. Panait and S. Luke. Cooperative Multi-Agent Learning: The State of the Art. *JAAMAS*, 11(3):387–434, 2005.

[16] J. Pineau, M. Montemerlo, M. Pollack, N. Roy, and S. Thrun. Towards Robotic Assistants in Nursing Homes: Challenges and Results. In *RAS Special Issue on Socially Interactive Robots*, 2003.

[17] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. Ros: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.

[18] S. Rosenthal, M. Veloso, and A. Dey. Learning Accuracy and Availability of Humans who Help Mobile Robots. In *Twenty-Fifth Conference on Artificial Intelligence (AAAI)*, San Francisco, USA, August 2011.

[19] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online Planning Algorithms for POMDPs. *JAIR*, 32:663–704, 2008.

[20] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, New Jersey, USA, 2003.

[21] P. E. Rybski, A. Larson, H. Veeraraghavan, M. LaPoint, and M. Gini. Communication strategies in Multi-Robot Search and Retrieval: Experiences with MinDART. In *Symposium on Distributed Autonomous Robotic Systems*, 2004.

[22] M. Sridharan, J. Wyatt, and R. Dearden. Planning to See: A Hierarchical Aprroach to Planning Visual Actions on a Robot using POMDPs. *Artificial Intelligence*, 174:704–725, 2010.

[23] S. Zhang, M. Sridharan, and X. Li. To Look or Not to Look: A Hierarchical Representation for Visual Planning on Mobile Robots. In *International Conference on Robotics and Automation*, 2011.

What am I doing? Automatic Construction of an Agent's State-Transition Diagram through Introspection

Constantin Berzan
Department of Computer Science
Tufts University
Medford, MA 02155, USA
constantin.berzan@tufts.edu

Matthias Scheutz
Department of Computer Science
Tufts University
Medford, MA 02155, USA
mscheutz@cs.tufts.edu

ABSTRACT

Infrastructures for implementing agent architectures are currently unaware of what tasks the implemented agent is performing. Such knowledge would allow the infrastructure to improve the agent's autonomy and reliability. For example, the infrastructure could detect abnormal system states, predict likely faults and take preventive measures ahead of time, or balance system load based on predicted computational needs. In this paper we introduce a learning algorithm to automatically discover a state-transition model of the agent's behavior. The algorithm monitors the communication between architectural components, in the form of function calls, and finds the frequencies at which various functions are polled. It then determines the states according to what polling frequencies are active at any time. The two main novel features of the algorithm are that it is completely *unsupervised* (it requires no human input) and *task-agnostic* (it can be applied to any new task or architecture with minimal effort).

Categories and Subject Descriptors

I.2.9 [**Artificial Intelligence**]: Robotics; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation

Keywords

introspection, state-transition model, unsupervised

1. INTRODUCTION

The architectures of robotic agents are often implemented in some middleware or software infrastructure [7]. The infrastructure's purpose is to abstract over hardware details and provide various advanced services to the architecture, such as automatic distribution of components over different computational resources, location-independent service discovery, communication with remote components, and various others. Infrastructures might have mechanisms to monitor their distributed network of components (e.g. to automatically restart crashed components) [10]. But infrastructures do not "know" what tasks the implemented agent is performing. Such knowledge could improve an agent's reliability and autonomy, especially in long-term sustained operations. For example, the infrastructure could detect abnormal system states, predict likely faults and take preventive measures ahead of time, or balance system load based on predicted computational needs.

The challenge is to obtain the knowledge needed to predict the agent's behavior. One possibility is for the designer to explicitly represent all possible system states. This is clearly difficult even for fairly small systems, such as a robot that performs navigation tasks. Moreover, such a description might fail to specify how the agent reacts to environmental contingencies, such as the appearance of an obstacle. Hence, in addition to the overall description of the system, some kind of learning component is necessary to integrate information about how environmental factors influence the agent's behavior.

Since the infrastructure would need an online learning component anyway, the other possibility would be for the infrastructure to *discover* the entire operation of the implemented agent, without any need to specify abstract system behavior or relevant system states. This means that the infrastructure would have to extract state information from the agent's internal, *subjective* perspective only, since it does not have access to any external, objective information such as the agent's global coordinates.

In this paper we introduce an unsupervised agent-centric learning algorithm to automatically discover a model of the agent's behavior. The algorithm monitors the communication patterns among architectural components, and builds a state diagram that reflects the agent's task model. We start with some background on related approaches for learning behavioral models. We then introduce our proposed approach, and demonstrate its operation in several robotic tasks implemented in the ADE infrastructure [10]. We show that the state diagrams generated automatically from the agent's internal perspective can nicely match state diagrams created manually from an external observer's perspective. We then discuss properties and shortcomings of the proposed method, provide a summary of our contributions, and outline future steps for improving results and performing larger-scale evaluations.

2. BACKGROUND

Conventional methods for modeling the behavior of an autonomous agent are based on observing the agent from

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

an external perspective. This *external approach* is based on methods used by ethologists in describing animal behavior [5]. First, we observe the agent and determine a set of low-level actions that it performs. For a rodent, these could include resting, walking, grooming, eating, and drinking [6]. Once the low-level action repertoire is defined, we describe the environmental conditions that cause the agent to switch between actions. This way we obtain a state diagram of the agent's behavior, where the states are the low-level actions, and the transitions are the observable conditions that cause action switches.

The behavior of an autonomous agent is typically represented as some variation of a Hidden Markov Model (HMM). If the set of states is known in advance and we have a training sequence of observations, we can learn the parameters of a HMM using algorithms such as Baum-Welch [1, 9]. For example, Guillory et al. [5] learn a model of the agent's behavior using Input/Output HMMs. Their approach requires as input a set of possible perceptions for the agent, and human-labeled example trajectories. Similarly, Delmotte and Egerstedt [2] learn simple control programs from externally-observed data. Goldberg and Mataric [4] present an algorithm for learning Augmented Markov Models (AMMs), which are similar to HMMs, but allow each state to produce a single symbol. Their algorithm takes a sequence of symbols and processes it online to update an initially empty AMM.

The strength of the external approach is that the state labels are directly meaningful to human beings, because the model was built based on observed behavior. On the downside, the external approach always requires some amount of domain-specific knowledge, such as a model of the agent's perceptions, or labels for an execution trajectory. Labeling can be difficult when the observer's action abstraction does not directly correspond to the agent's internal action representation. For example, a robot may use several different actions to follow a corridor, depending on its current goal. But an external observer might treat all these actions as the same. Similarly, an external observer might discriminate among multiple actions (e.g. approach wall, turn away from wall) even though the agent's control system does not discriminate among them (e.g. because the agent uses a potential-based approach for traversing hallways). As a result, the state-transition diagram will contain states that have no counterpart in the agent control system. Furthermore, the external approach treats the agent like a black box, and will miss unobservable state changes (such as perceiving a door and storing its location for future exploration). The external approach will also miss any behaviors exhibited outside of the observation period. And, in general, it might not be possible to observe the agent in certain situations (e.g. a cleaning robot in the sewage system).

To overcome some of these complications we turn to the *internal approach*, where the observer is the infrastructure in which the agent control system is implemented. Unlike any external observer, the infrastructure has exact information about component interactions. The agent is no longer a black box. On the other hand, the infrastructure usually has no information about the functional role of these components in the agent architecture. Wallace [12] discusses the advantages of self-assessment (internal monitoring) as opposed to external monitoring for detecting runtime errors. Other unsupervised approaches for acquiring a model of the world and self are being explored in the field of developmental robotics [13, 11]. Our work is most similar to the robot-introspection work of Fox et al. [3], which learns a HMM of the robot's behavior from raw sensor data. They start with a set of human-labeled states, which they then refine. Our method requires no labeling whatsoever for model construction, but only for evaluation. Furthermore, the variables and sensory features used by Fox et al. have to be chosen by hand for each task, whereas the log data we use requires no human pre-processing. Also, Fox et al. require multiple runs of the same task to learn the model, whereas our method allows building a model from a single run.

The ADE infrastructure mediates communication between the agent's components. During task execution we can record the interaction between components, in the form of function calls. The patterns in the recorded call log can be used to define states, which can then be organized in a state-transition model. These states will reflect the agent's control system, and as discussed above, they will not necessarily correspond to states described by an external observer. In general, the best we can hope for from the internal approach is that it will come reasonably close to an external model built from human observations. We are looking for a middle ground between a model that is overly specific (too many states) and one that is overly general (too few states). If this goal can be achieved, the internal approach will have several advantages over the external approach. First, internal modeling does not require any human labeling effort. This means that it can easily be applied to new tasks or architectures, and it can, in principle, run online while the robot is performing its task. Second, the model corresponds closely to the actual control flow of the robot, so there is no risk of "cheating" by imposing structure that is not really there (which an external observer might be tempted to do). Finally, and most importantly, the robot itself knows what state it is in, and it can use its own model to make predictions. This can enable the robot to balance load or to predict failures before they happen.

3. METHOD

Our state extraction algorithm is based on the following key observation: Throughout the execution of the task, the architecture polls various functions at regular intervals. For example, while going through a hallway, the robot might poll the `checkMotion` function, but not the `getLaserReadings` function. When entering a door, the robot might poll `getLaserReadings`, but not `checkMotion`. We associate each distinct polling pattern with a state. The states detected this way are grounded solely in subjective data collected internally by the robot.

To explain our method, we use a short example task, where the robot turns inside a room, and then exits through the door and into the hallway. The input data is an execution log, containing a list of function calls. Each call is identified by a time stamp, a component and function name, and a set of arguments (which we currently ignore). Figure 2a shows some sample log entries.

We first determine the possible *polling frequencies* for each function. Then, we identify the time intervals when each polling frequency is active (we call these the *instances* of a given polling frequency). We define a *state* as a set of active polling frequencies. We use the detected instances to determine the states, and the state-transition history. Finally,

we prune the state-transition history to remove superfluous states, and we use the pruned transition history to construct a state diagram. We proceed to describe this process in greater detail, together with the parameters that each step requires. It is helpful to follow Figure 2 while reading the rest of this section.

3.1 Polling frequencies

We start by computing the time difference between pairs of consecutive calls to the same function. If there are n calls to function `f`, we get $n-1$ values, which we call the *cadence values* for `f`. If some caller polls `f` every 100ms, we expect to see a cluster of cadence values around 100ms. Figure 2b shows a histogram plot of the cadence values for our example task. We can see clusters for `getLaserReadings`, `checkMotion`, `updateMoveToRel`, and `getPlan`. To extract these clusters, we make a single-linkage hierarchical clustering[1] of the cadence values for each function. We then put two cadence values in the same flat cluster if the difference between them is less than a threshold D_{max}. If a cluster is supported by less than N_{min} cadence values, we discard it. Figure 2c shows the resulting clusters (the clusters are ranges of values, and we identify them by their center). In general, there can be more than one cluster for any given function. We refer to a function (e.g. `getLaserReadings`) together with one of its clusters (e.g. 327ms) as a *polling frequency* (e.g. `getLaserReadings` at 327ms).

This step has the following parameters:

- D_{max}: maximum difference between adjacent cadence values for them to belong to the same cluster
- N_{min}: minimum number of cadence values to constitute a polling frequency

3.2 Instances of each polling frequency

In the next step, we want to find all *instances* when a given polling frequency is active. For example, for the polling frequency "`f` at 200ms," we want to find all time intervals $(t_{\text{begin}},\ t_{\text{end}})$ when `f` is being polled at 200ms. To find these intervals, we sweep over the logged calls in the order they occurred, keeping track of which polling frequencies are possible at each point. If over an interval of time, we see at least C_{min} calls to `f`, and the time difference between each consecutive pair of calls is within a tolerance T of 200ms, we save an instance for this polling frequency. Since calls that are slightly off-time happen often, we forgive an early or late call, if the next call is on time. Figure 2d shows a timeline of the logged calls. Figure 2e shows the detected instances for each polling frequency.

This step has the following parameters:

- C_{min}: minimum number of calls to constitute an instance
- T: tolerance with respect to a polling frequency (for a call to be counted towards an instance)

3.3 States and the state diagram

We define a *state* to be a set of active polling frequencies. We extract states from the instances detected in the previous step. At each point in time, a given set of polling frequencies is active, and if this set does not correspond to an existing state, it becomes a new one. Figure 2f shows the states and state-transition history discovered this way. Because instances of different polling frequencies are never perfectly aligned, this process results in a lot of superfluous states, in which very little time is spent. We therefore prune the state-transition history, discarding all state visits shorter than V_{min}. The time spent in a discarded state is redistributed to the previous and next state in the transition history. Figure 2g shows the states and state-transition history after pruning.

This step has a single parameter:

- V_{min}: minimum time spent in a state (for pruning)

Finally, we use the pruned state-transition history to build a state diagram of the task, shown in Figure 1. We indicate the start state with an arrow, and the final state with a double border.

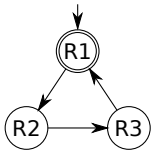


Figure 1: The state diagram obtained for the example task.

We use this simple way of constructing a diagram to illustrate the results of our state-extraction method. The state-transition history obtained in the final step (Figure 2g) is simply a sequence of states, which we could use to train a HMM or AMM if desired.

4. EXPERIMENTAL RESULTS

Evaluation is one of the main difficulties of robot introspection. Because the learned model is grounded in the robot's subjective observations, there is no "ground truth" that we can use for a direct comparison. Instead, as pointed out by Fox et al. [3], we are forced to compare the learned model with a human observer's *interpretation* of what the robot actually did. We cannot escape the limitation that interpretations may differ across observers.

We evaluate our algorithm by looking for bisimilarity between the constructed state diagram D_R and another state diagram D_O, representing a human observer's interpretation of the robot's behavior. Bisimilarity (or bisimulation) is a relationship between two state-transition systems that behave in the same way, in the sense that one system simulates the other and vice-versa. Park [8] provides a technical definition in the context of automata theory.

We have tested our state-extraction algorithm on three different tasks, using the ADE simulator. In the Boxes task, the robot moves objects from several source boxes into a destination box. In the Hallway task, the robot traverses a hallway looking for wounded people in every open room. The Combined task is a composition of the two tasks above. The robot traverses a hallway and performs the Boxes task inside each room.

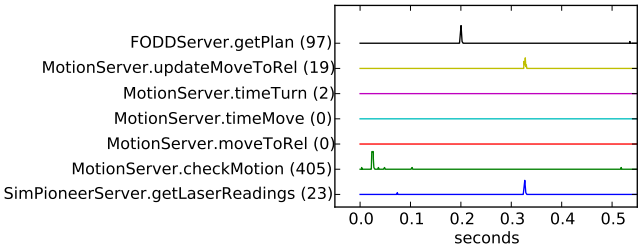
We used the following parameters in our experiments: $D_{max} = 5$ms, $N_{min} = 10$, $C_{min} = 3$, $T = 20$ms, $V_{min} = 1.3$s. For each task, we performed the following steps:

1. Run the task, generating the log and recording a screencast of the simulator window.

[1]We also tried k-means, RANSAC, and Gaussian Mixture Model fitting, but the simple hierarchical clustering worked best.

system time		component name	function name	arguments
1306509565603	CALL:	com.motion.MotionServer	moveToRel	0.0020581365076 0.890346846488
1306509565878	CALL:	com.adesim.SimPioneerServer	getLaserReadings	
1306509573546	CALL:	com.motion.MotionServer	checkMotion	1306509573522

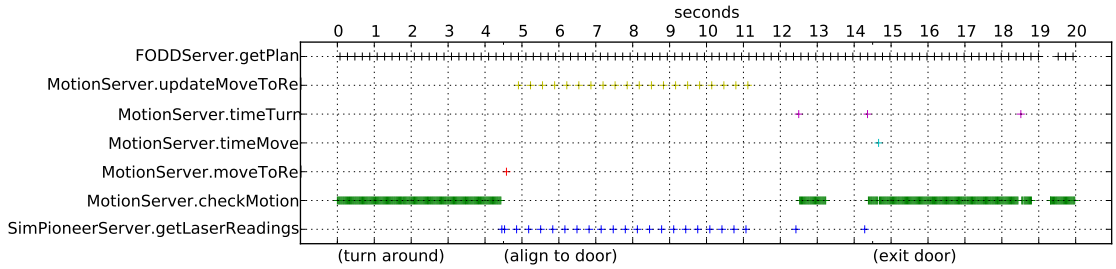
(a) Sample log entries



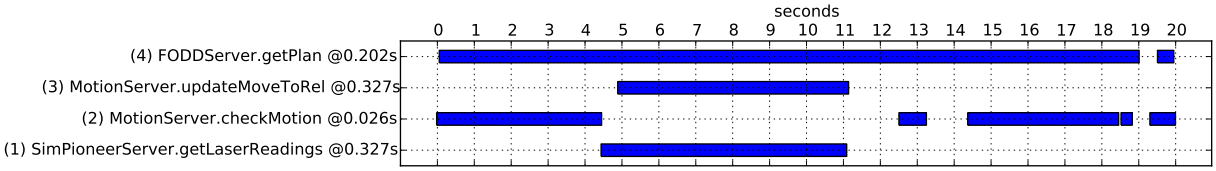
(b) Cadence histogram. The number of cadence values for each function is shown in parentheses.

function	polling frequencies
getLaserReadings	327ms
checkMotion	26ms
updateMoveToRel	327ms
getPlan	202ms

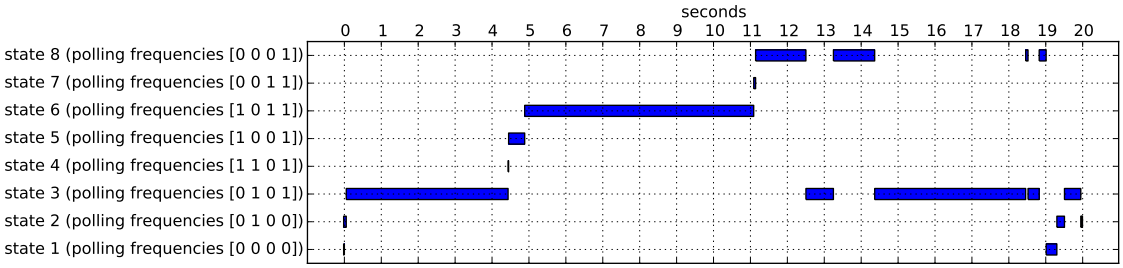
(c) Detected polling frequencies



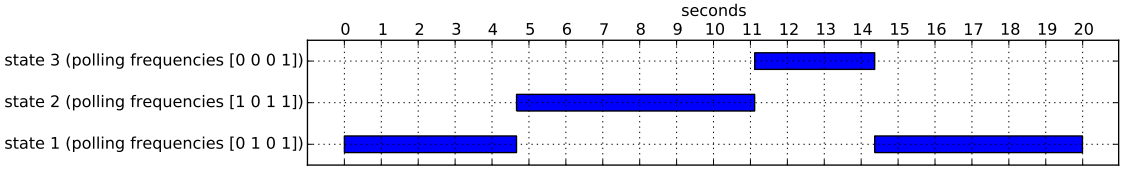
(d) Timeline of function calls, with the observed events shown on the bottom (the algorithm does not use these labels)



(e) Detected instances for each polling frequency



(f) States and state-transition history before pruning. The states are identified by a set of active polling frequencies. For example, state 3 is marked [0 1 0 1], which means the second (`checkMotion` at 26ms) and fourth (`getPlan` at 202ms) polling frequencies are active.



(g) States and state-transition history after pruning

Figure 2: Our method, described in section 3. From the log, we obtain the cadence values (b), which we cluster to obtain the polling frequencies (c). We then traverse the log (d), find the instances of each polling frequency (e), determine states based on what polling frequencies are active (f), and finally prune the state-transition history (g).

2. Watch the screencast and create a state diagram based on the behavior we observe. We call this the observer's state diagram, D_O.
3. Run the state-extraction algorithm on the robot's log, obtaining an unlabeled state diagram.
4. Label the state diagram given by the algorithm, by watching the screencast and observing the robot's behavior during each of the detected states. We call the result the robot's state diagram, D_R.
5. Create an expanded state diagram, D_E, such that every state in D_E corresponds to exactly one state in D_O, and exactly one state in D_R. (We view D_O and D_R as two different ways to decompose the robot's behavior into states, and D_E as their common denominator.)
6. Show that D_R and D_O are bisimilar, by defining a function f mapping states in D_O to states in D_E, and a function g mapping states in D_R to states in D_E.

4.1 The Boxes Task

In this task, the robot is in a room with a destination box and several source boxes, which may be empty or contain a single object. The robot first does a 360-degree sweep to determine the location of the boxes. Then it visits each source box, and if it finds an object inside, it carries the object to the destination box. We stop the run after the robot visits four source boxes. Figure 3 depicts the robot's environment and trajectory.

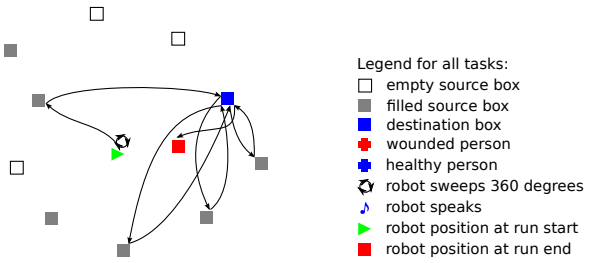


Figure 3: Map of the Boxes task, with legend.

The robot's state diagram D_R has two states, depicted in Figure 6a, and the observer's state diagram D_O has six states, shown in Figure 6b. For this task, the expanded state diagram D_E is identical to D_O. The full state-transition histories for D_R and D_O are given in Figure 6c. The bisimilarity between D_R and D_O is given by:

$$f = \{R_1 \mapsto \{O_1, O_3, O_5\}, R_2 \mapsto \{O_2, O_4, O_6\}\}$$

4.2 The Hallway Task

In this task, the robot traverses a hallway with several rooms. Upon seeing an open door, the robot enters the room and does a 360-degree sweep, looking for wounded people. If it finds a wounded person, the robot pauses and sends a spoken message to the operator. It then exits the room and continues traversing the hallway. We stop the run after the robot visits three rooms. Figure 4 depicts the robot's environment and trajectory. (Please consult the legend on the right side of Figure 3.)

The robot's state diagram D_R has six states, depicted in Figure 7a. The observer's state diagram D_O has five states, depicted in Figure 7c. The expanded state diagram D_E has eleven states, shown in Figure 7b. The full state-transition histories for D_R, D_E, and D_O are given in Figure 7d. The bisimilarity between D_R and D_E is given by:

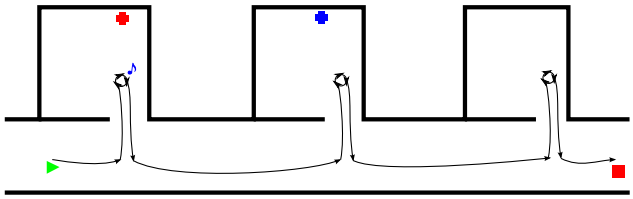


Figure 4: Map of the Hallway task.

$$f = \{R_1 \mapsto \{E_1\}, R_2 \mapsto \{E_2\}, R_3 \mapsto \{E_3\}, R_4 \mapsto \{E_4, E_9\}, R_5 \mapsto \{E_5, E_7, E_{10}\}, R_6 \mapsto \{E_6, E_8, E_{11}\}\}$$

and the bisimilarity between D_O and D_E is given by:

$$g = \{O_1 \mapsto \{E_1, E_2\}, O_2 \mapsto \{E_3, E_4, E_5, E_6\}, O_3 \mapsto \{E_7\}, O_4 \mapsto \{E_8\}, O_5 \mapsto \{E_9, E_{10}, E_{11}\}, \}$$

4.3 The Combined Task

This task is a composition of the Hallway and Boxes tasks. The robot traverses a hallway with several rooms. Upon seeing an open door, the robot enters the room and performs the Boxes task: It does a 360-degree sweep to find any boxes, and carries any objects from source boxes to the destination box. When finished (or if no source boxes are found), the robot exits the room and continues traversing the hallway. The run starts with the robot sweeping the first room, and it ends after the robot exits the third room. Figure 5 depicts the robot's environment and trajectory. (Please consult the legend on the right side of Figure 3.)

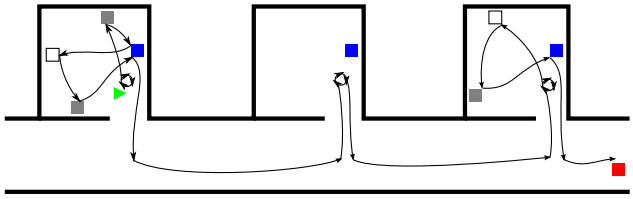


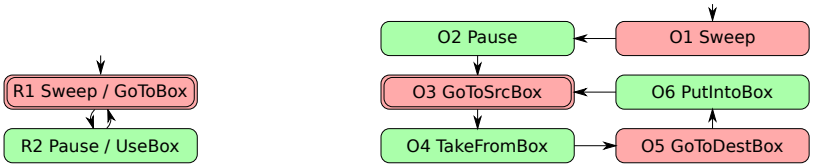
Figure 5: Map of the Combined task.

The robot's state diagram D_R has six states, depicted in Figure 8a. The observer's state diagram D_O has ten states, depicted in Figure 8c. The expanded state diagram D_E has sixteen states, shown in Figure 8b. The full state-transition histories for D_R, D_E, and D_O are given in Figure 9. The bisimilarity between D_R and D_E is given by:

$$f = \{R_1 \mapsto \{E_2, E_4, E_6, E_9, E_{15}\}, R_2 \mapsto \{E_1, E_3, E_5, E_7, E_{10}, E_{16}\}, R_3 \mapsto \{E_8, E_{14}\}, R_4 \mapsto \{E_{11}\}, R_5 \mapsto \{E_{12}\}, R_6 \mapsto \{E_{13}\}\}$$

and the bisimilarity between D_O and D_E is given by:

$$g = \{O_1 \mapsto \{E_1\}, O_2 \mapsto \{E_2\}, O_3 \mapsto \{E_3\}, O_4 \mapsto \{E_4\}, O_5 \mapsto \{E_5\}, O_6 \mapsto \{E_6\}, O_7 \mapsto \{E_7\}, O_8 \mapsto \{E_8, E_9, E_{10}\}, O_9 \mapsto \{E_{11}, E_{12}\}, O_{10} \mapsto \{E_{13}, E_{14}, E_{15}, E_{16}\}\}$$

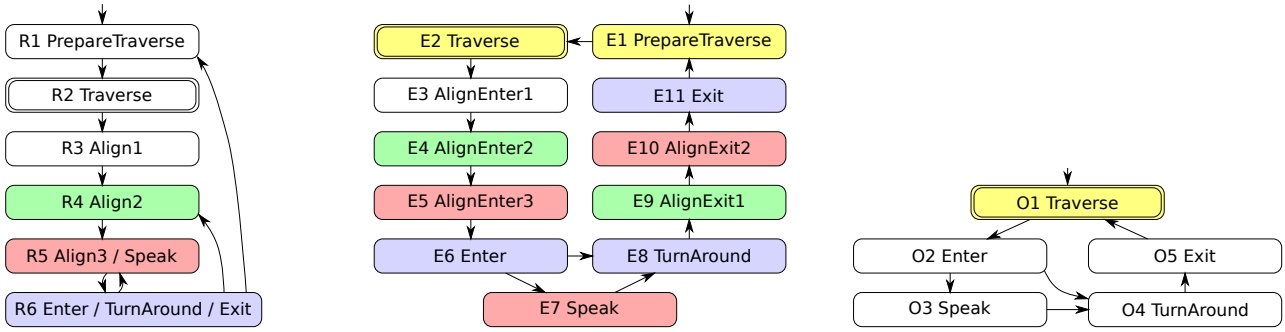


(a) D_R: robot's state diagram (b) D_O: observer's state diagram

R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1
O_1	O_2	O_3	O_4	O_5	O_6	O_3	O_4	O_5	O_6	O_3	O_4	O_5	O_6	O_3	O_4	O_5	O_6	O_3

(c) State-transition history in D_R (top) and D_O (bottom). Time moves from left to right (not drawn to scale).

Figure 6: State diagrams for the Boxes task. The start state has a loose arrow coming into it. The end state has a double border. To illustrate bisimilarity, the states corresponding to R_1 are shaded in red, and the states corresponding to R_2 are shaded in green.



(a) D_R: robot's state diagram (b) D_E: expanded state diagram (c) D_O: observer's state diagram

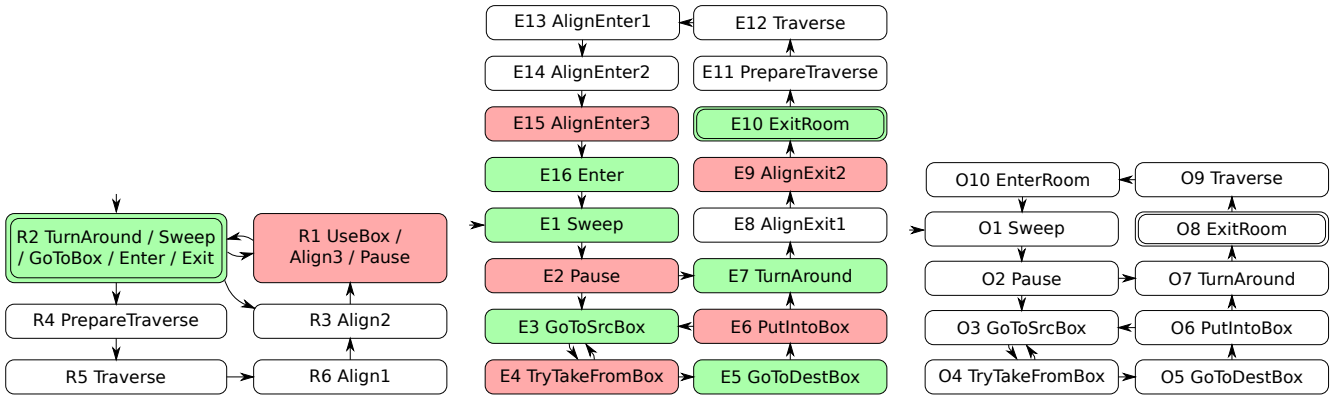
R_1	R_2	R_3	R_4	R_5	R_6	R_5	R_6	R_4	R_5	R_6	R_1	R_2	R_3	R_4	R_5	R_6	
E_1	E_2	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}	E_1	E_2	E_3	E_4	E_5	E_6	E_8
O_1		O_2				O_3	O_4	O_5			O_1		O_2				O_4

...

R_4	R_5	R_6	R_1	R_2	R_3	R_4	R_5	R_6		R_4	R_5	R_6	R_1	R_2			
E_9	E_{10}	E_{11}	E_1	E_2	E_3	E_4	E_5	E_6	E_8	E_9	E_{10}	E_{11}	E_1	E_2			
O_5			O_1		O_2				O_4	O_5			O_1				

(d) State-transition history in D_R (top), D_E (middle), and D_O (bottom). Time moves from left to right (not drawn to scale).

Figure 7: State diagrams for the Hallway task. We illustrate bisimilarity by shading some states in D_R and D_O, and their corresponding states in D_E (R_4 in green, R_5 in red, R_6 in blue, and O_1 in yellow).



(a) D_R: robot's state diagram (b) D_E: expanded state diagram (c) D_O: observer's state diagram

Figure 8: State diagrams for the Combined task. We illustrate bisimilarity by shading R_1 in red and R_2 in green, showing the corresponding states in D_E.

R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_3	R_1	R_2	R_4	R_5	R_6	R_3	R_1	R_2		
E_1	E_2	E_3	E_4	E_3	E_4	E_5	E_6	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}	E_{11}	E_{12}	E_{13}	E_{14}	E_{15}	E_{16}	E_1	...
O_1	O_2	O_3	O_4	O_3	O_4	O_5	O_6	O_3	O_4	O_5	O_6	O_7	O_8			O_9		O_{10}				O_1	

...	R_1	R_2	R_3	R_1	R_2	R_4	R_5	R_6	R_3	R_1	R_2		R_1	R_2	R_1	R_2	R_1	R_2	R_1	R_2	R_3	R_1	R_2
	E_2	E_7	E_8	E_9	E_{10}	E_{11}	E_{12}	E_{13}	E_{14}	E_{15}	E_{16}	E_1	E_2	E_3	E_4	E_3	E_4	E_5	E_6	E_7	E_8	E_9	E_{10}
	O_2	O_7	O_8			O_9		O_{10}				O_1	O_2	O_3	O_4	O_3	O_4	O_5	O_6	O_7	O_8		

Figure 9: State-transition history for the COMBINED task, in D_R (top), D_E (middle), and D_O (bottom). Time moves from left to right (not drawn to scale).

5. DISCUSSION

We start by discussing how the extracted state diagrams relate to the real world. As noted before, the only information used by our algorithm is the function-call log given by the infrastructure. This means that the algorithm has no knowledge of the *meaning* of a function call, and so cannot derive the meaning of a state, either. Consider the state diagram D_R extracted by the algorithm for the HALLWAY task (Figure 7a). We can imagine a state diagram that has the same number of states, the same arrows, and the same start and end state. But if the states have completely different labels, the semantics of the state diagram would also be radically different. This is the price we pay for being completely agent-centric: Our algorithm has no way to get at the semantics of states.

Despite this limitation, we were able to show how the extracted state diagrams D_R are bisimilar to state diagrams made by an external observer D_O. We did this by building a more fine-grained ("expanded") state diagram D_E, and showing that D_R and D_O are both bisimilar to D_E. This correspondence is encouraging, because it shows that the states detected introspectively often correspond to distinct observable behaviors. Note that bisimilarity is a rather strong requirement, and that we would need something weaker if some state transitions were not detected (e.g. if we usually detect $R_a \to R_b \to R_c$, but sometimes we just detect $R_a \to R_c$).

Comparing the extracted state diagrams with those made by an external observer illustrates some limitations of introspection. For example, in the BOXES task our algorithm was unable to distinguish between the PAUSE, TAKEFROMBOX, and PUTINTOBOX states, and lumped the three together in R_2 (Figure 6a). The reverse can also happen: In the HALLWAY task, our algorithm split the observer's single state ENTER into the sequence $R_3 \to R_4 \to R_5 \to R_6$. This shows that parts of the extracted state diagram can be either more coarse-grained or more fine-grained than what an observer sees.

We now turn to a discussion of our method. Our original idea for extracting states was to slide a fixed-size observation window over the call log, count the number of calls to each function in each observation, and cluster the resulting vectors. This turned out to work very poorly, because the fixed-size window introduced discretization errors, states with a short time span tended to be ignored, and observations taken at the transition between two states gave rise to spurious clusters. Moreover, the number of clusters had to be known in advance. We developed our current method after we realized that most functions were being polled at regular intervals, and that we could take advantage of the polling patterns to distinguish between states. (One could presumably design an architecture that performs no polling whatsoever; in that case our algorithm would need to be adapted.)

Our method has five parameters: two for finding the polling frequencies, two more for detecting the instances of each polling frequency, and one for pruning the state-transition history. For our experiments, we set the parameters by hand, noticing how they affect the algorithm's behavior on the three tasks. For example, if D_{max} or T are too small, then we would most likely miss slow polls (e.g. one call every second). If they are too large, then we might detect spurious polling frequencies and instances. If N_{min} or C_{min} are too large, we might miss states that the robot visits for only a short time. If they are too small, then we would most likely have many spurious state transitions. Finally, V_{min} has to strike a balance between pruning too much (joining states that should be distinct), and pruning too little (leaving spurious states). We believe that the optimal setting for these parameters depends on variables in the infrastructure, such as the call latency.

We briefly sketch the time complexity of each part of our algorithm. The hierarchical clustering to find polling frequencies takes $O(N_c^2 \log N_c)$ time and $O(N_c^2)$ space, where N_c is the number of calls to a given function. Therefore, for N_f, functions that are being polled, finding all polling frequencies takes $O(N_f \cdot N_c^2 \log N_c)$ time. After that, assuming N_{pf} polling frequencies were detected, finding the instances naively takes $O(N_{pf} \cdot N_c)$ time. For N_i instances, extracting the states takes $O(N_i)$ time. If the resulting state-transition history has N_t transitions, pruning it takes $O(N_t^2)$ naively, and can be improved to $O(N_t \log N_t)$ by using a heap.

The initial clustering of cadence values appears to be the bottleneck. In our experiments, the cadence values had millisecond precision. To keep the running time reasonable, it was sufficient to reduce the number of duplicate cadence values to N_{min} (this does not affect the results). For convenience, we used single-linkage hierarchical clustering, and we then created flat clusters based on a distance threshold. It should be possible to replace this expensive part of the algorithm with simply sorting the cadence values, and then traversing them, finding the flat clusters directly, and avoiding the hierarchical clustering altogether. Such an approach would take $O(N_c \log N_c)$ time per function. Based on this back-of-the-envelope analysis, we expect our algorithm to scale well with increasingly complex tasks.

Finally, we discuss how our method could help with fault detection and load balancing. (We plan to evaluate these proposals in future work.) Suppose the robot performs a task for which it has already built a state diagram. If it encounters a previously unseen state transition, the robot could signal to the operator that something unexpected is happening. Using introspection alone, the robot would be unable to distinguish between a fault (in which case it should

state	visit count	time spent mean	std
R_1	4	3.68 s	1.22 s
R_2	4	14.33 s	7.09 s
R_3	3	2.18 s	0.64 s
R_4	6	4.81 s	1.75 s
R_5	7	2.37 s	0.93 s
R_6	7	17.11 s	6.93 s

Table 1: Mean and standard deviation of the time spent in each state of the HALLWAY task.

notify the operator) and a new state (in which case it should update its internal model). The only way to clarify the situation is to ask the operator: "Am I doing something wrong?" or "Is this supposed to happen?" Spending too much or too little time in a state may also indicate a failure. To detect this, the robot could maintain statistics about the time spent in each state. Table 1 shows the statistics collected during the HALLWAY task. The standard deviations are high, indicating that the time spent in a state depends on environmental features (e.g. the length of hallway between two rooms), and not just on what the robot is doing. This suggests that applying the unmodified AMM-learning algorithm of Goldberg and Mataric [4] could be problematic, because a state with high variance would be split in two.

To perform load balancing, the robot needs to know what components of the architecture are active in each state. We can obtain this information directly from the logs. Knowing its current state and the possible next states, the robot can instantiate components on different hosts to achieve load balancing. It can also conserve energy by suspending a component if it is unlikely to be used in the near future.

6. CONCLUSIONS AND FUTURE WORK

The main contribution of this paper is an algorithm to extract a state diagram of an agent's behavior from the communication patterns of its architectural components. The algorithm monitors function calls between components, and finds the frequencies at which various functions are polled. It then determines the states according to what polling frequencies are active at any time. Unlike external approaches to modeling robot behavior, our algorithm is completely *unsupervised* (it requires no human input) and *task-agnostic* (it can be applied to any new task or architecture with minimal effort).

We evaluated the algorithm in three robotic example tasks. We demonstrated that the state diagrams extracted by the algorithm are bisimilar to state diagrams made by an external observer who watches the robot perform its task. We also discussed how our algorithm can be used for fault detection and load balancing.

The most immediate direction for future work is to make the algorithm work online, which would allow the robot to learn during task execution and take advantage of what it has learned so far. To turn the current version into an online algorithm, we would need to detect new polling frequencies and instances iteratively, without maintaining a full history of every call so far. Using the online version, we intend to demonstrate fault detection and load balancing based on an actual robot.

Another extension to our algorithm is to consider "rare" calls: occasional calls to functions that occur without polling. This should enable us to distinguish among states that are very similar otherwise (e.g. TAKEFROMBOX and PUTINTOBOX). We are also looking into extending our algorithm by incorporating sensor data and function-call arguments, and by building a probabilistic state-transition model, while remaining completely unsupervised and task-agnostic. Finally, it would be interesting to see how much better the state-transition model can get if we are allowed to occasionally ask the operator questions, such as "What am I doing right now?" or "Am I doing the same thing that I was doing a minute ago?".

7. REFERENCES

[1] E. Charniak. *Statistical Language Learning*. Language, Speech, and Communication. MIT Press, 1996.

[2] F. Delmotte and M. Egerstedt. Reconstruction of low-complexity control programs from data. In *43rd IEEE Conference on Decision and Control*, volume 2, pages 1460–1465, 2004.

[3] M. Fox, M. Ghallab, G. Infantes, and D. Long. Robot Introspection through Learned Hidden Markov Models. *Artificial Intelligence*, 170(2):59–113, 2006.

[4] D. Goldberg and M. J. Mataric. Augmented Markov Models, 1999.

[5] A. Guillory, H. Nguyen, T. Balch, and C. L. Isbell. Learning executable agent behaviors from observation. In *Proc. of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*, 2006.

[6] H. Jhuang, E. Garrote, X. Yu, V. Khilnani, T. Poggio, A. D. Steele, and T. Serre. Automated home-cage behavioural phenotyping of mice. *Nature communications*, 1(6):68, 2010.

[7] J. Kramer and M. Scheutz. Robotic development environments for autonomous mobile robots: A survey. *Autonomous Robots*, 22(2):101–132, 2007.

[8] D. Park. Concurrency and automata on infinite sequences. In P. Deussen, editor, *Theoretical Computer Science*, volume 104 of *Lecture Notes in Computer Science*, pages 167–183. Springer Berlin / Heidelberg, 1981. 10.1007/BFb0017309.

[9] L. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257 –286, feb 1989.

[10] M. Scheutz. ADE - Steps towards a distributed development and runtime environment for complex robotic agent architectures. *Applied Artificial Intelligence*, 20(4-5), 2006.

[11] D. Stronger and P. Stone. Towards autonomous sensor and actuator model induction on a mobile robot. *Connection Science*, 18(2):97–119, June 2006.

[12] S. A. Wallace. S-assess: a library for behavioral self-assessment. In *Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, AAMAS '05, pages 256–263, New York, NY, USA, 2005. ACM.

[13] J. Weng, J. McClelland, A. Pentland, O. Sporns, I. Stockman, M. Sur, and E. Thelen. Artificial intelligence. Autonomous mental development by robots and animals. *Science*, 291(5504):599–600, January 2001.

Learning from Demonstration with Swarm Hierarchies

Keith Sullivan
Department of Computer Science
George Mason University
Fairfax, VA 22030
ksulliv2@cs.gmu.edu

Sean Luke
Department of Computer Science
George Mason University
Fairfax, VA 22030
sean@cs.gmu.edu

ABSTRACT

We present a supervised learning from demonstration system capable of training stateful and recurrent collective behaviors for multiple agents or robots. A model space of this kind is often high-dimensional and consequently may require a large number of samples to learn. Furthermore, the inverse problem posed by emergent macrophenomena among multiple agents presents major challenges to supervised learning methods. Our approach reduces the size of the state space, and shortens the gap between individual behaviors and macrophenomena, by manually decomposing individual behaviors and arranging the agents into a tree hierarchy. This makes it possible to train potentially large numbers of agents using a small number of samples. We demonstrate our system using hundreds of agents in a simulated foraging task, and on real robots performing a collective patrolling task.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence—*Learning*

General Terms

Performance

Keywords

Learning from Demonstration, Multiagent Systems, Robotics

1. INTRODUCTION

Programming agent behaviors is a tedious, time consuming task involving multiple code, test, and debug cycles. Creating these behaviors requires significant programming ability, which makes training the agents attractive. One training approach is Learning from Demonstration (LfD), where the agent learns a behavior in real-time based on examples provided by a human demonstrator. LfD teaches an agent a policy which maps environmental features to agent actions.

Supervised learning methods are a natural fit for LfD, as the trainer is directly providing examples. But we note that supervised *cooperative multiagent training* has a surprisingly small literature. From an extensive survey of cooperative multiagent learning [15] it was found that only a small number of papers deal with supervised learning, and most of those are in the area of *agent modeling*, whereby agents learn about one another, rather than being trained by the experimenter. The lion's share of the remaining literature tends to fall into feedback-based methods such as reinforcement learning or stochastic optimization (genetic algorithms, etc.). For example, in one of the more well-known examples of multiagent layered learning [17], the supervised task ("pass evaluation") may be reasonably described as agent-modeling, while the full multiagent learning task ("pass selection") uses reinforcement learning. This is not unusual.

Why is this so? Supervised training, as opposed to agent modeling, generally requires that agents be told which micro-level behaviors to perform in various situations; but the experimenter often does not know this. He may only know the emergent macro-level phenomenon he wishes to achieve. This inverse problem poses a significant challenge to the application of supervised methods to such problems. The standard response to inverse problems is to use a feedback-based technique. But there is an alternative: to decompose the problem into sub-problems, each of which is simple enough that the gulf between the micro- and macro-level behaviors is reduced to a manageable size. This is our approach.

Our multiagent training method rests upon an LfD system we have developed called *Hierarchical Training of Agent Behaviors* (or HiTAB). In its basic form this system is a *single*-agent training system which learns a hierarchical finite state automaton (HFA) represented as a Moore machine. Individual states in the automaton either correspond to agent behaviors, or may themselves be another HFA. An HFA is constructed iteratively: using with a behavior library consisting solely of atomic behaviors (e.g., turn, go forward), the demonstrator trains an automaton describing a more complicated composed behavior, which is then saved to the behavior library. The now expanded behavior library is again used to train a more abstract and capable automaton, which is likewise saved to the library. This process continues until the desired behavior is trained.

Our goal is to apply this training technique not just to single agents but to supervised training of teams and swarms of arbitrary size. Our approach is as follows. We organize the agents into an *agent hierarchy*, a tree structure where leaf nodes are the individual agents or robots performing tasks, and non-leaf nodes are (possibly virtual) *controller agents*. After we have trained and distributed individual behaviors to the leaf-node agents using HiTAB, we the group them into small, manageable teams (perhaps of size five), each headed

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

by a controller agent. We then train the controller using HiTAB in much the same way that the individual agents were trained: but his states dictate the collective behaviors of his small team. After we have trained the small team, we group controller agents together in teams, each such team headed by a higher-level controller agent. This training and grouping continues until the entire swarm has been organized into a hierarchy.

This tree-structured organization fits between fully decentralized ("swarm"-style) multiagent systems and fully centralized systems. While the tree structure has obvious disadvantages (e.g., it is not robust to agent failure), it has one overriding scaling advantage: regardless of its size, at any position in the structure an agent must deal only with a fixed number of agents (his superior and immediate subordinates). We are taking advantage of this to make the multiagent training task feasible regardless of the size of the swarm. At all times we are training a controller to direct a small number of agents (his immediate subordinates), regardless of the position of the controller in the hierarchy. The micro-to-macro gulf is much smaller and simpler with five or fewer agents than it it is with hundreds or thousands of agents. Furthermore, the use of HFAs at the controller level allows us to decompose complex team behaviors into simpler ones in much the same way that we simplified the problem in the single-agent HiTAB case.

Hierarchies are a natural fit for organizing heterogeneous agent swarms, but interestingly they're also useful for swarms of agents with homogeneous behaviors too. In this paper we show this: we train hierarchies of behaviors to control *homogeneous* agents, and demonstrate trained behaviors which are superior to those found in flat ("swarm"-style) structures.

2. RELATED WORK

Agent Hierarchies. Hierarchies have long been employed to control a robot programmatically, from the traditional multi-tier planner/executive/control hierarchical frameworks, to *behavior hierarchies* establishing precedence among competing robot behaviors, of which an early example is the Subsumption architecture [2]. A significant body of literature has constructed groups of agents, with each agent employing its own internal hierarchical behavior mechanism [16, 5, 21]. Hierarchies *among* agents are less common, for example [6]. Some recent literature has focused on *hierarchies of control* among heterogeneous agents [8]. Hierarchies may also be constructed dynamically as a mechanism for task allocation [12].

Learning from Demonstration. Much of the learning from demonstration literature may be divided into systems which learn plans (for example [14, 20]) and those which learn *policies*, that is, stateless mappings from the agent's feature vector to a desired action [1, 4, 10, 13]. Some work involves stateful models related to ours, notably via Hidden Markov Models. For example, [9] treat states not as behaviors but as hidden world conditions which the learner is attempting to discover and optimize for. [7] learns transitions between states corresponding to behaviors, though it does not label the transitions.

Multiagent Learning. One of the primary challenges addressed by this paper is in applying learning from demonstration — at its heart a supervised task — to the multiagent case. As noted in [15], supervised learning methods are not very common in multiagent learning: by far the lion's share of literature is based on reward-based methods such as reinforcement learning or stochastic optimization. Of those supervised methods, many fall in the category of *agent modeling*, where agents learn about one another rather than about a task given to them by demonstrator. For example, in the celebrated [18], the supervised task ("pass evaluation") is reasonably described as agent modeling, while the full multiagent learning task ("pass selection") uses reinforcement learning. Multiagent learning may also be achieved via confidence estimation rather than reinforcement learning [3]. An alternative way to bridge the macro-micro gulf is to eliminate the macrobehaviors entirely by issuing separate micro-level training directives each individual agent [11, 19]. We argue that this approach is unlikely to scale.

3. AGENT BEHAVIOR TRAINING

HiTAB develops behaviors in the form of hierarchical finite-state automata (HFA), where each state in an automaton is either an atomic behavior in the agent, or another automaton. Multiple states in the automaton may map to the same atomic behavior or lower-level automaton. As the objective is to enable learning from demonstration with a minimum number of samples, the trainer first defines the behavior by manually decomposing it into a hierarchy of sub-behaviors. For each sub-behavior, he then selects the *features* of the environment and the *states* needed to learn the behavior. HiTAB thus learns only the *transition functions* from each state within the HFA. These simplifications, made possible by decomposition, radically reduce the dimensionality of the problem and enable learning on a much smaller number of samples. Formally, the HFA is a tuple $\langle S, B, F, T \rangle \in \mathcal{H}$:

- $S = \{S_1, \ldots, S_n\}$ is the set of *states* in the automaton. Included is one special state, the *start state* S_0, and zero or more *flag states*. Exactly one state is active at a time, designated S_t.

  The purpose of a flag state is simply to raise a flag in the automaton to indicate that the automaton believes that some condition is now true. Two obvious conditions might be *done* and *failed*, but there could be many more. Flags in an automaton appear as optional features in its *parent* automaton. For example, the *done* flag may be used by the parent to transition away from the current automaton because the automaton believes it has completed its task.

- $B = \{B_1, \ldots, B_k\}$ is the set of *basic behaviors*. Each state is associated with either a basic behavior or *another automaton* from $\mathcal{H}$, though recursion is not permitted.

- $F = \{f_1, \ldots, f_m\}$ is the set of observable *features* in the environment. At any given time each feature has a numerical value. The collective values of F at time t is the environment's *feature vector* $\vec{f_t} = \langle f_1, \ldots, f_m \rangle$.

- $T = \vec{f_t} \times S \to S$ is the *transition function* which maps the current state S_t and the current feature vector $\vec{f_t}$ to a new state S_{t+1}.

- Optional free variables (parameters) $G_1, \ldots, G_n$ for basic behaviors and features generalize the model: each behavior B_i and feature f_i are replaced as $B_i(G_1, \ldots, G_n)$ and $f_i(G_1, \ldots, G_n)$. The evaluation of the transition function

and the execution of behaviors are based on ground instances of the free variables. For example, rather than have a behavior called *go to the ball*, we can create a behavior called *goTo(A)*, where A is left unspecified. Similarly, a feature might be defined not as *distance to the ball* but as *distanceTo(B)*. If such a behavior or feature is used in an automaton, either its parameter must be bound to a specific *target* (such as "the ball" or "the nearest obstacle"), or it must be bound to some higher-level parent of the automaton itself. Thus HFAs may themselves be parameterized.

Single-Agent Training. Training is an iterative process between a *training mode* and a *testing mode.* In the training mode, the agent performs exactly those behaviors as directed by the demonstrator. During training, each time the demonstrator chooses a new behavior, the agent records a training example: a tuple $\langle S_t, \vec{f}_t, S_{t+1} \rangle$ which stores the current feature vector, along with the states corresponding to the old and new behaviors. If state S_{t+1} must be executed exactly once, then no additional examples are recorded. Otherwise, a default example is stored: $\langle S_{t+1}, \vec{f}_t, S_{t+1} \rangle$, which tells the agent to continue in the current state if the given feature vector is observed again. The feature vector is specified by the user from a library of predefined but parameterizable features selected for the behavior.

Once enough examples are collected, the demonstrator switches to the *testing mode*, which causes the agent to learn the transition functions within the finite-state automaton. For a given state S_i, HiTAB takes all examples of the form $\langle S_i, f_t, S_j \rangle$ and reduces them to $\langle f_t, S_j \rangle$. HiTAB then applies a classification algorithm to these examples, using the f_t as data and S_j as their labels. At present HiTAB uses decision trees with probabilistic leaf nodes for our classifiers.

After all the transition functions are built, the agent begins performing the learned behavior. If the demonstrator observes the agent performing an incorrect behavior, he may issue corrections, causing the agent to switch back to training mode and collect additional examples, then reenter testing mode with revised training functions. This continues until the demonstrator is satisfied with the behavior, and saves it to the behavior library. At this point, unused states and features are trimmed from the automaton, and any parameterized behaviors and features are bound to a target (e.g., "nearest obstacle"), or to a parameter of the automaton itself. The behavior is now available as a state for training another, higher-level HFA.

Multiagent Training. Once we have trained a library of useful individual behaviors using HiTAB, how might we extend this to training collective multiagent behavior? The obvious (distributed) approach is to simply endow all agents with the same top-level behavior. An alternative centralized approach is to define a single master *controller agent* in charge of all subsidiary agents. The subsidiary agents all have the same behaviors in their libraries; but the controller agent has its own separate library of behaviors, both basic behaviors and learned automata. A controller agents' basic behaviors do not manipulate the controller, but instead correspond to a unique behavior in the libraries of the subsidiaries. When a controller agent transitions to a new basic behavior, this directs the subsidiaries to immediately start performing the corresponding behavior in their libraries.

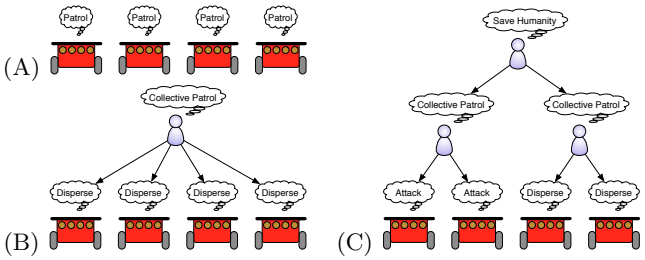


Figure 1: Three notions of homogeneity. (A) Each agent has the same top-level behavior, but acts independently. (B) The top-level behavior all agents is the same, but may all be switched according to a higher-level behavior under the control of a controller agent. (C) Squads in the team are directed by different controller agents, whose behaviors are the same but may all be switched by a higher-level controller agent (and so on).

Our framework is in-between: we define a hierarchy of controller agents. The basic agents are grouped into subgroups, each headed by a level-1 controller agent; then various level-1 controller agents are grouped as subsidiaries to level-2 agents, and so on, up to level-m agents forming one or more roots. Just as all basic agents have the same behaviors, all controller agents at a given level have the same behaviors. The actual structure of the hierarchy (number of levels, number of agents per controller, etc.) is pre-defined by the user. Depending the configuration of the hierarchy, this framework can range from fully distributed to fully centralized, with many points in-between.

The agents are trained starting at the leaf nodes, and working towards the root node one level at a time. After training basic agents in the usual fashion, we may then train a level-1 controller agent, then a level-2 controller agent, and so on. All agents at a given level within the hierarchy run the same HFA, but at any time they may be in different states of that HFA. controller agent training is essentially the same as for basic agents: the user directs the controller agent to perform various behaviors, which in turn cause the controller's subsidiaries to perform behaviors. This adds examples to a database from which transitions are learned.

While the basic behaviors for a controller agent are straightforward, what is a controller agent's set of features? We presume that, unlike a basic agent, a controller agent isn't embodied: his features are derived from statistical results from his subsidiaries: for example "a basic agent in my group is stuck (or isn't)", or "all my immediate subsidiaries are 'done' (or not)", or "the average Y position of basic agents in my group". Typically a controller agent only accesses its immediate subsidiary agents, but there are no restrictions as to how deep in the hierarchy the controller agent can gather information. Like an agent's basic features, the choice of features available to a controller agent are domain-specific.

We ultimately plan to use this method to develop heterogeneous team behaviors: but for now we are concentrating on homogeneous behaviors. We note that this embedding of the HFA training into an agent hierarchy suggests at least three different notions of "homogeneous" behaviors, as shown in Figure 1. First, all agents may simply perform the exact same HFA, but independent of one another. But we can go

Figure 2: Learned multi-robot behavior. Demonstrator is holding a green ("intruder") target.

further than this and still stay within the aegis of homogeneity: we may add a controller agent which controls *which* HFA the agents are performing. It does so by running its own HFA with those subsidiary HFA as basic behaviors. Coordination may continue further up the chain: second- or higher-level controller agents may also dictate their subsidiaries' choice of HFAs.

4. ROBOT DEMONSTRATION

We begin with a simple demonstration which illustrates this approach on actual robots, using a simple hierarchy of four Pioneer robots under the control of a single controller agent. We trained this group to perform a pursuit task while also deferring to and avoiding a "boss". Each robot had a color camera and sonar, and was marked with colored paper. The boss, intruders to pursue, and a home base were also marked with paper of different colors. (See Figure 2).

The task was as follows. Ordinarily all agents would *Disperse* in the environment, wandering randomly while avoiding obstacles (by sonar) and each other (by sonar or camera). Upon detecting an intruder in the environment, the robots would all *Attack* the intruder, servoing towards it in a stateful fashion, until one of them was close enough to "capture" the intruder and the intruder was eliminated. At this point the robots would all go to a home base (essentially *Attack* the base) until they were *all* within a certain distance of the base. Only then would they once again *Disperse*. At any time, if the boss entered the environment, each agent was to *Run Away* from the boss: turn to him, then back away from him slowly, stopping if it encountered an obstacle behind.

This task was designed to test and demonstrate every aspect of the hierarchical learning framework: it required the learning of hierarchies of individual agent behaviors, stateful automata, behaviors and features with targets, both continuous and categorical features, multiple agents, and learned hierarchical behaviors for a controller agent.

Each robot was provided the following simple basic behaviors: to continuously go *Forwards* or *Backwards*, to continuously turn *Left* or *Right*, to *Stop*, and to *Stop* and raise the *Done* flag. Transitions in HFAs within individual agents were solely based on the following simple features: whether the current behavior had raised the *Done* flag; the minimum value of the *Front Left*, *Front Right*, or *Rear* sonars; and the *X Coordinate* or the *Size* of a blob of color in the environment (we provided four colors as targets to these two features, corresponding to *Teammates*, *Intruders*, the *Boss*, and the *Home Base*). Each robot was dressed in the *Teammate* color.

We began by training agents to learn various small parameterized HFAs, as detailed in Figure 3, Subfigures 1 through 7. Note that the *Servo* and *Scatter* HFAs are stateful: when the target disappeared, the robot had to discern which direction it had gone and turn appropriately. Since our system has only one behavior per state, we enabled multiple states with the same behavior by training the trivial HFAs in subfigures 3A through 3D. Training these behaviors required approximately 30 minutes.

We then experimented with the "basic" homogeneous behavior approach as detailed in Figure 1(A): each agent simply performing the same top-level behavior but without any controller agent controlling them. This top-level behavior was *Patrol* (Figure 3, Subfigure 8), and iterated through the three previously described states: dispersing through the environment, attacking intruders, and returning to the home base. We did not bother to add deferral to the "boss" at this point.

Coordinated Homogeneity. Simple homogeneous coordination like this was insufficient. In this simple configuration, when an agent found an intruder, it would attack the intruder until it had "captured" it, then go to the home base, then resume dispersing. But other agents would not join in unless they too had discovered the intruder (and typically they had not). Furthermore, if an agent captured an intruder and removed it from the environment, other agents presently attacking the intruder would not realize it had been captured, and would continue searching for the now missing intruder indefinitely!

These difficulties highlighted the value of one or more controller agents, and so we have also experimented with placing all four robots under the control of a single controller that would choose the top-level behavior each robot would perform at a given time. The controller was trained to follow the *CollectivePatrol* behavior shown in Figure 3, Subfigure 9. This HFA was similar to the *Patrol* behavior, except that robots would attack when *any* robot saw an intruder, would all go to the Home Base when *any* robot had captured the intruder, and would all resume dispersing when *all* of the robots had reached the Home Base. This effectively solved the difficulties described earlier.

We provided the controller with three simple features: whether any robot had seen the Intruder's color; whether any robot was *Done*, and whether all robots were *Done*. We trained a simple hierarchical behavior on the controller agent, called *CollectivePatrolAndDefer* (Subfigure 10). We first added a new statistical feature to the controller agent: whether anyone had seen the Boss color within the last $N-10$ seconds. The controller agent would perform *CollectivePatrol* until someone had seen the Boss within the last 10 seconds, at which point the controller agent would switch to the *RunAway* behavior, causing all the agents to search for the Boss and back away from him. When no agent had seen the Boss for 10 seconds, the controller would resume the *CollectivePatrol* behavior (Figure 2).

Summary. This is a reasonably comprehensive team behavior, with a large non-decomposed finite-state automaton, spanning across four different robots acting in sync. We do not believe that we could train the agents to perform a behavior of this complexity without decomposition, and certainly not in real-time. There are too many states and aliased states, too many features (at least 12), and too many transition conditions. However decomposition is straightfor-

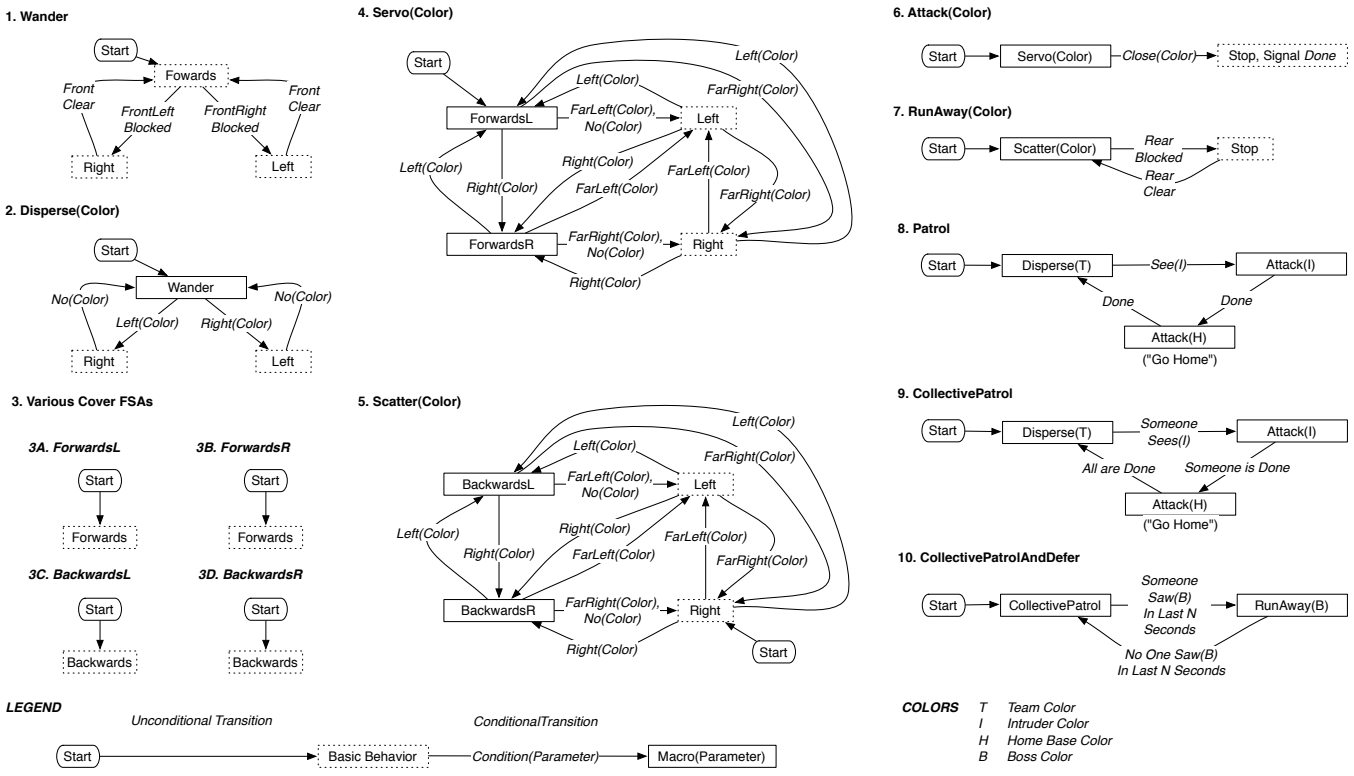


Figure 3: Decomposed hierarchical finite-state automaton learned in the demonstration. See discussion in the text on each subfigure. Most behaviors form a hierarchy within an individual robot, but *CollectivePatrol* and *CollectivePatrolAndDefer* form a separate hierarchy within the team controlling agent. Though the transition condition descriptions here are categorical sounding, most are in fact derived from continuous values: for example, *Left(Color)* is trained based on *X* coordinates of the color blob in the field of view.

ward into simple, easily trained behaviors with small numbers of features and states, simple (indeed often trivial) and easily trained transition functions, and features and states which may vary from behavior to behavior.

5. SIMULATION EXPERIMENTS

After conducting the robot demonstration above, we proceeded to conduct simulation experiments to quantify the benefit of controller agents, particularly as the hierarchy grew from a single controller to multiple levels of controllers. We applied our multiagent homogeneous hierarchies to a simulated box foraging problem: agents hunt for boxes, then pull them to a known deposit location. The boxes are randomly distributed throughout the environment, and after collection at the deposit, the box disappears and a new box is placed randomly in the environment. The environment consists of various circular "boxes" of different sizes, which likewise require different numbers of agents (5, 25, 125) to pull them.

We performed experiments involving *swarms* of independent agents, *groups* of agents under a single layer of controllers (called Level 1 controllers), and groups of agents under multiple layers of controllers (Level 1, Level 2, and so on). To perform these experiments required training three kinds of behaviors. First, we trained behaviors for each basic agent, then we trained behaviors for Level 1 controllers (designed to control basic agents), and finally we trained behaviors for Level N≥2 controllers (designed to control other controllers). This set of behaviors was sufficient to scale to any number of levels. We now describe the basic behaviors and features, and trained decompositions.

Basic Agent Behavior Decomposition. We decomposed and trained a basic agent's behavior hierarchy as follows:

- Agents' basic features were *DistanceTo(X), DirectionTo(X), ICanSeeABox, IAmAttachedToABox,* and *Done*. The first two features were parameterizable to either visible boxes or to the deposit location. The last feature was true when the *done* flag had been raised. Boxes could only be seen if they were within 10 units.
- Agents' basic behaviors were *Forward, RotateLeft, RotateRight, GrabBox, ReleaseBox, ReleaseBoxAndFinish,* and *Done*. Both *ReleaseBox* and *Done* would raise the *done* flag and (as normal) immediately transfer to the start state. *ReleaseBoxAndFinish* would as well, except that it would also raise a *finished* flag in the agent which could be detected by controllers as a feature. Boxes could only be grabbed if they were sufficiently close (5 units).
- Using *Forward, RotateLeft,* and *RotateRight*, we trained *Wander*, which wandered randomly.
- Using *Forward, RotateLeft,* and *RotateRight*, plus the *DistanceTo(X)* and *DirectionTo(X)* features, we trained the behavior *Goto(X)*, which servoed to a given target.

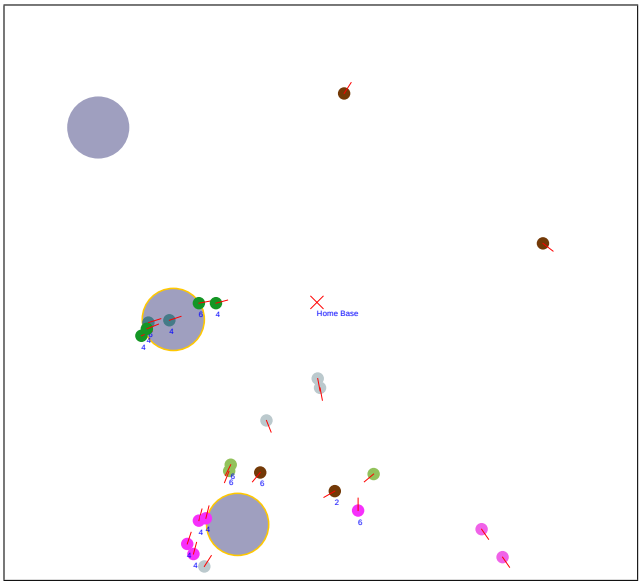


Figure 4: A screenshot of our system in action (showing part of the environment). The large grey circles are the boxes, and the X in the middle is the collection location. Note that while the agents pulling the box on left are all from the same subgroup, the box in the bottom is being pulled by agents from different subgroups.

- Using *Goto(X)*, *GrabBox*, *ReleaseBoxAndFinish*, and *DistanceTo(X)*, we trained *ReturnWithBox*, which pulled the box back to the deposit location and released it when the agent was close enough to home.
- Using *Wander* and *ReturnWithBox*, we trained *Forage*, a simple top-level composition which foraged for boxes and brought them to the deposit.

If agents were acting on their own (they had no controller), their top-level behavior would be simply *Forage*. When acting under a Level 1 controller, the current behavior of the agent would be determined by the controller. Training the agent's behavior required approximately 30 –40 minutes.

Level 1 Controller Agent Behavior Decomposition. A Level 1 controller's behavior hierarchy was as follows:

- A controller's basic features were *SomeoneIsFinished* and *SomeoneIsAttachedToABox*. The latter feature was true if any subsidiary agent had raised its *finished* flag. A controller also had access to an additional target: *closest-attached-agent*, which pointed to the subsidiary agent which had grabbed the box (if any).
- A controller's basic behaviors corresponded to the full set of behaviors of its subsidiary agents: *Forward, RotateLeft, RotateRight, GrabBox, ReleaseBoxAndDone, Done, Wander, Goto(X), ReturnWithBox,* and *Forage.*
- Using *Goto(closest-attached-agent)*, *ReleaseBox*, *ReturnWithBox*, *Forage*, *SomeoneIsAttachedToABox*, and *SomeoneIsFinished*, we trained the behavior *ControlForage*, which directed agents to *Forage* until an agent found a box.

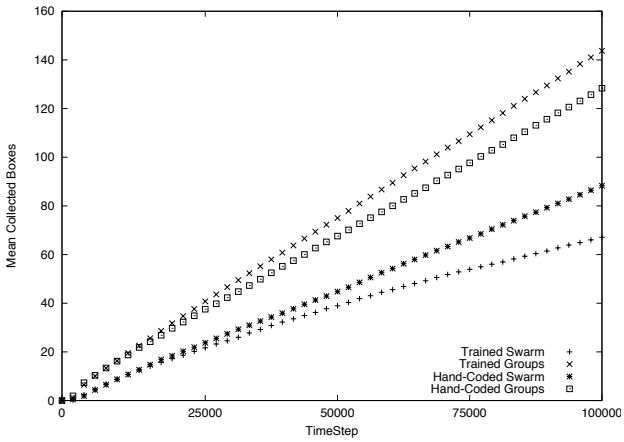


Figure 5: Mean number of boxes collected over time for the first experiment.

Then, the controller would direct agents to *Goto(closest-attached-agent)*; once agents where close to the attached agent, they would grab the box and begin pulling it towards the deposit location. Once one agent finished pulling the box, the controller would direct the agents to *ReleaseBox*, and to resume *Forage*. We also trained trivial *ReturnWithBox* and *Goto(X)* behaviors which simply called their corresponding basic behaviors.

If the agents were acting on their own (they had no Level 2 controller), their top-level behavior would be simply *ControlForage*. When acting under a Level 2 controller, the current behavior of the Level 1 controllers would be determined by their Level 2 controllers. Training Level 1 controller agents required a few minutes.

Level N≥2 Controller Agent Behavior Decomposition. All controller agents at levels ≥ 2 used exactly the same behavior hierarchy, which was:

- A controller's basic features were *SomeoneIsFinished* and *SomeoneNeedsHelp*. The former feature is true if a subsidiary agent knows of a box which requires more agents to push it than are available to the subsidiary agent. A Level N≥2 controller also had an additional target: *biggest-attached-agent*, which is the agent attached to the largest box that the N≥2 controller "knows about". The controller would learn of such boxes from its superior, or from subsidiary controllers unable to manage the box themselves.
- A Level N≥2 controller's basic behaviors corresponded to behaviors from its subsidiary controllers: *ControlForage*, *ReturnWithBox*, *Goto(X)*.
- We trained a version of *ControlForage* similar to the Level 1 *ControlForage* behavior. The difference is that the Level N≥2 behavior directs agents to *Goto(biggest-attached-agent)* when a Level 1 controller requires help. We also trained trivial *ReturnWithBox* and *Goto(X)* behaviors which simply called their corresponding basic behaviors. Training level N ≥ 2 controller agents required a few minutes for each level.

A Level N≥2 controller's top behavior was *ControlForage*.

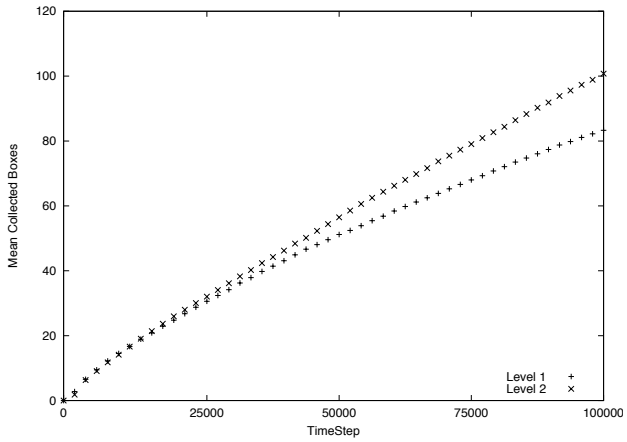


Figure 6: Mean number of boxes collected over time for the second experiment.

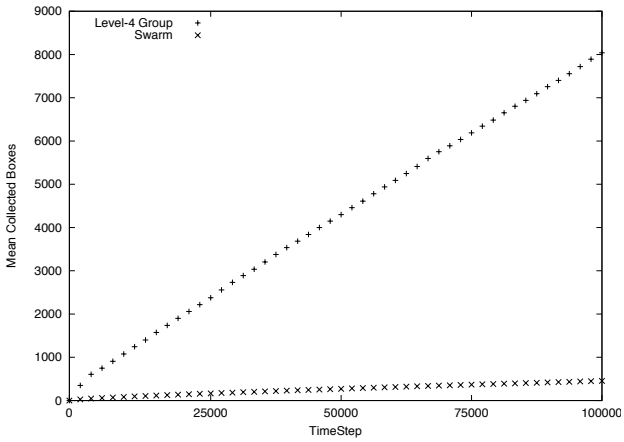


Figure 7: Mean number of boxes collected over time for the third experiment.

5.1 Experiments

For the first two experiments, we considered three hierarchical structures: (1) 50 independent agents (2) ten independent Level 1 controller agents, each heading a five-agent subgroup (3) two independent Level 2 controller agents, each heading five Level 1 controller agents, each heading a five-agent subgroup. These structures roughly correspond to the notions illustrated in Figure 1.

In the first experiment, we sought to demonstrate that a simple hierarchy can out-perform a group of independent agents; and additionally, that the behaviors learned in this experiment would perform adequately compared to fine-tuned hand-coded behaviors. In the second experiment, we sought to demonstrate that a two-layer hierarchy could outperform a one-layer hierarchy. In these experiments, the environment was 200×200 units and agents moved 0.1 units per timestep. Agents started at uniformly randomly distributed locations.

Finally, we tried a scalability experiment, comparing a different, even larger hierarchy against 625 independent agents.

Each experimental run lasted 100,000 timesteps, and each treatment had 100 independent runs. Treatments were gauged based on the mean number of boxes returned. Differences in results were measured at the final timestep with a 95% confidence, using Bonferroni-corrected two-tailed t-tests.

First Experiment: 1-Level Hierarchies. We began by comparing an entirely distributed *swarm* of 50 agents against a *group* of ten controller agents, each in charge of five basic agents. For each of these configurations, we performed runs using a set of trained behaviors and using a set of hand-coded behaviors. Figure 5 shows the results of all four sets of runs.

Independent Agents Versus Hierarchies: We expected the controller agents to outperform a distributed swarm due to the semi-centralized coordination available, because the controller enabled specific groups of agents to work together on a single box. Without controller agents, agents could become stranded at boxes waiting for other agents to help pull. These waiting agents simply relied on random discovery of the box by other agents to gather enough helpers. Figure 5 verifies the expected improvement due to the controllers. The improvement was statistically significant in both cases.

Hand-Coded Versus Trained: We then compared the trained versions of the two previous structures with hand-coded versions of the same. Figure 5 again shows the results. We had expected the hand-coded solutions to perform better, since trained solutions contained significant training error. But in fact, in the Level 1 hierarchy case, the trained solution actually performed statistically significantly better than the hand-coded solution! This was due to a more random exploration strategy which allowed agents to disperse throughout the environment better. This same exploration strategy didn't fare as well in the swarm case, however, because this strategy resulted in too many agents distributed across multiple boxes rather than pulling on the same box. While the results do not present a clear advantage to either training or programming, they do suggest that training the agents will crucially not *significantly* impair performance.

Second Experiment: 2-Level Hierarchies. We then compared the same Level 1 hierarchy as before against a two-level hierarchy: two Level 2 controllers, each in charge of five Level 1 controllers, each in charge of five agents. We changed the scenario to favor two levels of coordination: the environment now had eight boxes which each required five agents to pull, and two boxes which each required twenty-five agents to pull. Just as the first experiment was constructed so as to demonstrate the value of some degree of homogeneous coordination, the second experiment is meant to show the value of homogeneous coordination at two levels.

As shown in Figure 6, two levels significantly outperformed a single level, and for similar reasons as the first experiment. If in the one-level case a group discovered a 25-agent box, 4 other groups had to randomly discover the box before it could be moved and all the groups freed. But with two layers of coordination, we could train agents to work together not only in 5-agent groups but also in 25-agent groups.

Third Experiment: Large Numbers of Agents. Finally, we reran the first experiment using a *four*-level hierarchy: a single Level 4 controller in charge of five Level 3 controllers, each in charge of five Level 2 controllers, each in charge of five Level 1 controllers, each in charge of five agents. This arrangement results in 625 agents and 156 controller agents.

We compared this against a swarm of 625 independent agents. With the larger number of agents, we expanded the environment to 225×225, and provided the environment with 25 size-5 boxes, five size-25 boxes, and one size-125 box.

As would be expected, this was no contest: the swarm of agents were simply outclassed, as shown in Figure 7. This somewhat unfair contest was not intended to show the *efficacy* of the hierarchy, but simply that it this approach is capable of scaling to large numbers of agents and more complex environments.

6. CONCLUSIONS

This paper demonstrates a novel approach to the challenging task of multiagent learning from demonstration. The approach makes progress against the inherent inverse problem by performing macro-behavior decomposition throughout a swarm hierarchy, to the point that the differences between the micro-level and macro-level behaviors are small enough to be surmounted. It also applies behavior decomposition on the individual agent level, enabling a variety of tricks to significantly reduce the complexity and dimensionality of the learning space, so as to get by on a very small number of samples.

Though the obvious target for hierarchies in swarms is heterogeneous behaviors (and that is our first task as future work), the demonstrations here show that hierarchies may still be of benefit in the homogeneous behavior case. In this case, such hierarchies provide an alternate method of consistent, learnable coordination among agents.

We note that, from a machine learning perspective, the *individual* learned behaviors shown here are often quite simple. But this is exactly the point. Our goal is to enable rapid agent and robot behavior development. From this perspective, decomposition of a very complex joint model into many simple models promises to allow even novices to build multiagent behaviors rapidly because the number of samples need not be large.

7. REFERENCES

[1] D. C. Bentivegna, C. G. Atkeson, and G. Cheng. Learning tasks from observation and practice. *Robotics and Autonomous Systems*, 47(2-3):163–169, 2004.

[2] R. Brooks. A robust layered control system for a mobile robot. *IEEE Journal Of Robotics And Automation*, RA-2:14–23, April 1986.

[3] S. Chernova. *Confidence-based Robot Policy Learning from Demonstration*. PhD thesis, Carnegie Mellon University, 2009.

[4] J. Dinerstein, P. K. Egbert, and D. Ventura. Learning policies for embodied virtual agents through demonstration. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1257–1252, 2007.

[5] E. H. Durfee and T. A. Montgomery. A hierarchical protocol for coordinating multiagent behaviors. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, pages 86–93, Boston, MA, USA, 1990. AAAI Press.

[6] D. Goldberg and M. J. Mataric. Design and evaluation of robust behavior-based controllers. In T. Balch and L. E. Parker, editors, *Robot Teams: From Diversity to Polymorphism*, pages 315–344. A. K. Peters, 2002.

[7] D. Goldberg and M. J. Mataric. Maximizing reward in a non-stationary mobile robot environment. *Autonomous Agents and Multi-Agent Systems*, 6:2003, 2002.

[8] R. Grabowski, L. E. Navarro-Serment, C. J. J. Paredis, and P. K. Khosla. Heterogeneous teams of modular robots for mapping and exploration. *Autonomous Robots*, July 2000.

[9] G. E. Hovland, P. Sikka, and B. J. McCarragher. Skill acquisition from human demonstration using a hidden markov model. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2706–2711, 1996.

[10] M. Kasper, G. Fricke, K. Steuernagel, and E. von Puttkamer. A behavior-based mobile robot architecture for learning from demonstration. *Robotics and Autonomous Systems*, 34(2-3):153–164, 2001.

[11] M. F. Martins and Y. Demiris. Learning multirobot joint action plans from simultaneous task execution demonstrations. In *Proceedings of Autonomous Agents and Multi-Agent Systems Conference (AAMAS)*, pages 931–938, 2010.

[12] J. McLurkin and D. Yamins. Dynamic task assignment in robot swarms. In *Robotics: Science and Systems Conference*, 2005.

[13] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47(2-3):79–91, 2004.

[14] M. N. Nicolescu and M. J. Mataric. A hierarchical architecture for behavior-based robots. In *The First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 227–233. ACM, 2002.

[15] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, 2005.

[16] L. Parker. ALLIANCE: An architecture for fault tolerance multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2), 1998.

[17] P. Stone and M. Veloso. Layered learning and flexible teamwork in robocup simulation agents. In M. Veloso, E. Pagello, and H. Kitano, editors, *RoboCup-99: Robot Soccer World Cup III*, volume 1856 of *Lecture Notes in Computer Science*, pages 65–72. Springer Berlin / Heidelberg, 2000.

[18] P. Stone and M. M. Veloso. Layered learning. In R. L. de Mántaras and E. Plaza, editors, *11th European Conference on Machine Learning (ECML)*, pages 369–381. Springer, 2000.

[19] B. Takács and Y. Demiris. Balancing spectral clustering for segmenting spatio-temporal observations of multi-agent systems. In *IEEE International Conference on Data Mining (ICDM)*, pages 580–587, 2008.

[20] H. Veeraraghavan and M. M. Veloso. Learning task specific plans through sound and visually interpretable demonstrations. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2599–2604. IEEE, 2008.

[21] T. Vu, J. Go, G. Kaminka, M. Veloso, and B. Browning. MONAD: a flexible architecture for multi-agent control. In *AAMAS 2003*, pages 449–456, 2003.

Autonomous Robot Dancing Driven by Beats and Emotions of Music

Guangyu Xia
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
gxia@andrew.cmu.edu

Junyun Tay*
Mechanical Engineering Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
junyunt@cmu.edu

Roger Dannenberg
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
rbd@cs.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
veloso@cmu.edu

ABSTRACT

Many robot dances are preprogrammed by choreographers for a particular piece of music so that the motions can be smoothly executed and synchronized to the music. We are interested in automating the task of robot dance choreography to allow robots to dance without detailed human planning. Robot dance movements are synchronized to the beats and reflect the emotion of any music. Our work is made up of two parts: (1) The first algorithm plans a sequence of dance movements that is driven by the beats and the emotions detected through the preprocessing of selected dance music. (2) We also contribute a real-time synchronizing algorithm to minimize the error between the execution of the motions and the plan. Our work builds on previous research to extract beats and emotions from music audio. We created a library of parameterized motion primitives, whereby each motion primitive is composed of a set of keyframes and durations and generate the sequence of dance movements from this library. We demonstrate the feasibility of our algorithms on the NAO humanoid robot to show that the robot is capable of using the mappings defined to autonomously dance to any music. Although we present our work using a humanoid robot, our algorithm is applicable to other robots.

Categories and Subject Descriptors

I.2.8 [**Problem Solving, Control Methods, and Search**]: Plan execution, formation, and generation

General Terms

Algorithms

Keywords

autonomous robot dancing, motion-emotion mapping, scheduling, real-time synchronization

*Junyun Tay is in the Carnegie Mellon University-Nanyang Technological University Dual PhD Programme.

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

1. INTRODUCTION

Dancing motions for robots are usually created by choreographers and designed for a particular piece of music. If the piece of music changes, the dance movements of the robot will have to be recreated. We are interested in automating the task of robot dance choreography by generating sequences of dance movements from a motion library. The automatically generated choreography should satisfy several goals. First, the choreography should be safe for performance. For example, it should not cause the robot to fall or break. Second, the choreography should reflect the emotional character of the music. Peaceful music should be choreographed differently from music that sounds angry. Third, the dance should be synchronized to the music. Finally the dance should not be deterministic. Even when the emotion and tempo of the music remain constant, the dance should contain interesting variations.

To address the goal of safety, we compose dances from sequences of motion primitives. A motion primitive is a sequence of keyframes (static poses), interpolated to form a continuous motion. The motion primitives are designed to be interesting and safe when performed in any sequence.

We represent emotion using a two-dimensional activation-valence emotion space, which is commonly used to describe emotional states. We create a large library of motion primitives, by dividing the joints of the NAO humanoid robot into 4 categories, where each category of joints can actuate independently. Given that there is a large number of motion primitives, we contribute another algorithm that uses labelled emotional static postures data, collected with the NAO humanoid robot, to estimate the activation-valence values for each motion primitive. Motion primitives are then selected to match the emotional state of the music.

To synchronize dance to the music, we use the fact that motion primitives can be executed with different durations. We adjust the duration of each motion primitive so that the duration will be an integer multiple of beats. In practice, as the movements on the NAO humanoid robot may not execute according to the planned schedule of motion primitives, we maintain synchronization with the music by adjusting the durations of motion primitives in real time to compensate for differences between the schedule and the actual execution.

To create interesting variations in the dance, we use a first-order Markov model to generate dances stochastically. States correspond to motion primitives. The state transition probabilities are designed to produce smooth motion sequences by favoring next states that begin with a keyframe near the final keyframe of the current state. The state transition probabilities also depend upon the current emotion in the music, such that at any given time, state transition probabilities will prefer states that reflect the current emotion in the music. Figure 1 summarizes the process we described. We demonstrate the feasibility of our algorithms on the NAO humanoid robot to show that the robot is capable of using the mappings defined to autonomously dance to any music. Our algorithms are applicable to other robots as well.

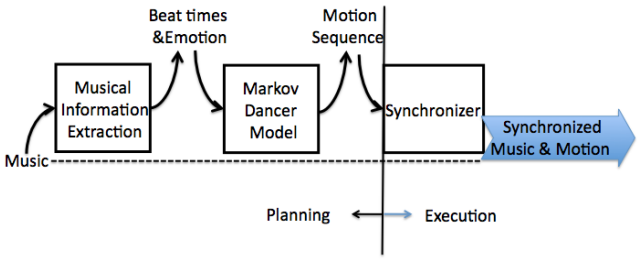


Figure 1: Summary of system diagram

2. RELATED WORK

Robot dances are generally preprogrammed by choreographers specifically for a piece of music. Researchers created motions for robots using motion capture data of humans dancing [13]. Our algorithm enables the robot to dance autonomously to any piece of music that is preprocessed using a library of predefined motions. Many explored emotional expressions of robots using the robot's facial features [2, 10] and did not consider using entire body postures and movements. Paul Ekman proposed 6 primary emotions: Happy, Sad, Angry, Surprised, Fear and Disgust [3], and we explore the use of body postures on humanoid robots to express 6 primary emotions. We use the information collected from these static emotion body postures to estimate the emotion of dance movements and organized them to reflect the emotions detected in music.

The creation of dancing characters is a common theme in computer animation research [8, 9, 11, 14, 15]. [11] uses a given sequence of motions to synthesize music, while others use given music to synthesize motion sequences. The work [14] focuses on a topological model of dance styles, while [8, 9, 15] focus on synthesizing motions according to musical beat times, which are very similar to our approach. Among them, only [15] synthesizes motions according to musical emotion. However their system uses only the intensity of music as an indicator of emotion. Our work enhances the use of beats by adopting a more recent and more accurate technique to identify beats and tempo. We also use a state-of-the-art emotion detection system coupled with our motion primitives for robots to convey richer emotions.

3. DANCE MOTIONS FORMALIZATION

We demonstrate our work on a NAO humanoid robot (Figure 2). The NAO robot has 21 joints and is a stand-alone autonomous robot with no facial features, except for LEDs in the eyes and ears. We group the joints into 4 categories:

1. Head (*Head*): HeadYaw, HeadPitch
2. Left Arm (*LArm*): LShoulderPitch, LShoulderRoll, LElbowYaw, LElbowRoll
3. Right Arm (*RArm*): RShoulderPitch, RShoulderRoll, RElbowYaw, RElbowRoll
4. Legs (*Legs*): LHipYawPitch, LHipRoll, LHipPitch, LKneePitch, LAnklePitch, LAnkleRoll, RHipRoll, RHipPitch, RKneePitch, RAnklePitch, RAnkleRoll

Figure 2: NAO humanoid robot

Each category of joints is defined to be $c = \langle J_{c,1}, \ldots, J_{c,|c|} \rangle$, where $c \in \{Head, LArm, RArm, Legs\}$ and $J_{c,1}, \ldots, J_{c,|c|}$ are the indices of the joints in the category. $|c|$ is the total number of joints in the category c. E.g., $Head = \langle 1, 2 \rangle$ where 1 is the index of HeadYaw and 2 is the index of HeadPitch.

3.1 Motion Primitive

Each keyframe is associated with a category c, and is defined as $K_c = \langle V_{c,1}, \ldots, V_{c,|c|} \rangle$, where $V_{c,j}$ contains the joint angle of joint index $J_{c,j}$. A motion primitive is $M_c(\beta) = \langle K_{c,1}, \beta D_1, K_{c,2}, \ldots, K_{c,F-1}, \beta D_{F-1}, K_{c,F} \rangle$ where F is the number of keyframes in M_c. D is the minimum time that it takes to move (interpolate) from one keyframe, $K_{c,f}$, to the next keyframe, $K_{c,f+1}$, and is pre-defined. We parameterize the motion primitive with β, where $\beta \in \mathbb{R}$ and $\beta \geq 1$. β is calculated using the beat times of the music so as to synchronize the motion primitive with the music (Section 5.2).

We assume independence of each body part by ignoring the effects of dynamics generated by motions. This categorization of joints according to body parts enables us to create a large variety of motion primitives, as we can create motion primitives for each category independently. We have a total of $8(Head) \times 9(LArm) \times 9(RArm) \times 26(Legs) = 16,848$ possible motion primitive combinations. The number of motion variations is actually much greater because motions primitives do not necessarily start and end synchronously. The 52 parameterized motion primitives are manually generated.

3.2 Schedule of Motion Primitives

A schedule of motion primitives belonging to the category c is defined as $S_c = \langle K_c, \mathcal{D}, M_{c,1}, I_1, M_{c,2}, \ldots, I_{p-1}, M_{c,p} \rangle$, where p is the total number of motion primitives in S_c. K_c contains the initial joint angles of the robot and $\mathcal{D}$ is the time to interpolate from K_c to the first keyframe of $M_{c,1}$. I_m is the time to interpolate from the last keyframe of $M_{c,m}$ to the first keyframe of $M_{c,m+1}$. We show how to calculate I_m in Section 5.2.1. We plan 4 schedules of motion primitives—$S_{Head}, S_{LArm}, S_{RArm}, S_{Legs}$ independently according to the emotions and beats of the music. Although the 4 schedules are generated independently, the robot can execute these 4 schedules simultaneously. The behaviour of the robot at a given point in time h is $B_h = \langle M_{Head}, M_{LArm}, M_{RArm}, M_{Legs} \rangle$

where M_c is the current motion primitive in S_c at time h where $c \in \{Head, LArm, RArm, Legs\}$.

Our formalization of the motion primitive, schedules of motion primitives and behaviour is general to use on different robots given the independence of the joints of the robot and that the joints can be actuated simultaneously.

4. MUSIC INFORMATION EXTRACTION

We obtain information directly from music audio signals to coordinate the robot dance motions with music. The extracted information consists of *emotions*, so that the robot's motions can be consistent with the mood of the music, and *beats*, which allow the robot to synchronize to musical pulses.

4.1 Emotion Extraction

It is well known that musical emotion has a significant impact on human dancers' movements. Our autonomous robot dance algorithm is driven by musical emotions.

We use SMERS [7], a state-of-the-art music emotion recognition system based on audio features and support vector regression (SVR). SMERS performs a forced classification into 11 emotion categories and achieved a 94% agreement with expert human labelers.

4.1.1 Emotion Representation

SMERS adopts Thayer's 2-dimensional Activation-Valence (AV) model [16] to represent musical emotion (Figure 3).

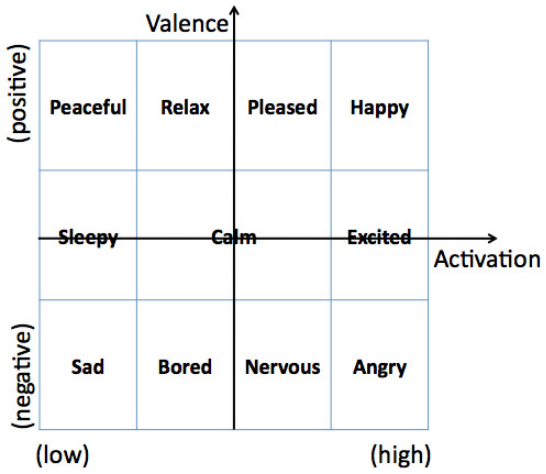


Figure 3: 2-dimensional emotion model

The emotion of a piece of music is represented by an AV value, (a, v) where a denotes Activation and v denotes Valence; $a, v \in \mathbb{R}, a \in [-1, 1], v \in [-1, 1]$. SMERS was developed and tested assuming each track of music (song) has a single main emotion. We consider that emotion may change over time within a piece, so we use a 30-second sliding window with a 15-second overlap. Therefore, music emotion is represented by a vector of (a, v) coordinates. The i^{th} element in the vector (indexing starts at 0) represents the emotion of the music at time $15i + 15$ seconds.

4.1.2 SVR training and decoding

Emotion labeling relies on SVR, which learns a mapping between feature vectors and emotion vectors. In our application, the feature vector $x_i \in \mathbb{R}^6$ is a vector of extracted music audio features, which include estimated key (one of 12 major or minor keys), average energy and standard deviation of energy, estimated tempo, standard deviation of beat duration, and harmonicity (see [7] for more details). The emotion vector is an (a, v) coordinate denoted by $y_i \in \mathbb{R}^2$. Since the original training data contains music audio with emotion labels such as Peaceful or Happy, we replace these labels with the (a, v) coordinates of the middle of the corresponding block as shown in Figure 3. Given a set of training data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$, SVR will try to find the optimal mapping function between input x_i and output y_i. In the decoding step, for a segment of music audio, we extract the feature vector x and apply the learned regression model to get the output (a, v) coordinates of the piece of music. These coordinates could be quantized to obtain a discrete label (e.g. Angry), but for choreography, we use the continuous numerical representation directly.

4.2 Beat Tracking

Musical beats reflect the basic period or pulse of music. It is typical for humans to synchronize the dance motions to the musical beats, so it is important for our algorithm to detect and track beats in music audio.

Much work has been done in the area of beat tracking. Currently, most algorithms are based on autocorrelation analysis or onset component detecting [1, 4, 5, 6]. Most recently, [1] proposed a beat tracking method that combines autocorrelation analysis and Neural Networks learning. The work done by Goto [6] is based on onset component and rhythm structure analysis. In nearly all cases, beat detection is based on some audio feature associated with note onsets and drum beats, such as change in amplitude or spectrum. Peaks in these features mark likely candidates for beat locations. Since musical beats mostly occur with an overall stable frequency (tempo), these candidate locations are filtered by looking for ones that are regularly spaced. Our approach estimates a global tempo (the overall beats per minute of a piece of music) by analyzing the autocorrelation of onset features throughout a piece, and then finds the best beat times by using dynamic programming [4]. This method performed well in the MIREX-06 evaluations [4] and generally works well when there are clear beats and steady tempo.

4.2.1 Global tempo estimation

The global tempo estimation algorithm is executed in three steps. First, the onset strength envelope (Figure 4) of the whole piece of music is calculated from a crude perceptual model (see [4] for more details). Second, the autocorrelation of the onset strength is computed. When the lag matches the beat or its multiples, the autocorrelation should be closer to 1. The result of one piece of music is shown in Figure 5. Third, the highest peak (ignoring the peak at 0) is detected and the corresponding lag is chosen as the global tempo.

4.2.2 Find best beat times

Local beat times correspond to perceived onsets in the audio signal of a piece of music. Given the global tempo, the beat times should not only be local onset peaks but should be equally spaced according to the global tempo. An objective function is defined to reflect these two goals.

$$S(\boldsymbol{T}) = \alpha \sum_{i}^{N} Onset(t_i) - (1 - \alpha) \sum_{i+1}^{N} dist(t_i - t_{i-1}, C) \quad (1)$$

Here, $\boldsymbol{T} = [t_1, t_2, \ldots, t_N]$ refers to the sequence of beat

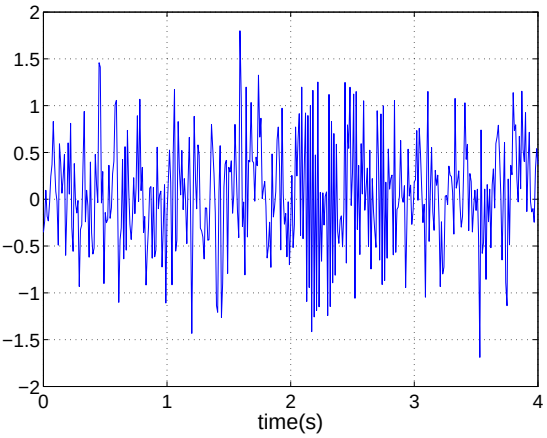


Figure 4: Part of the onset envelope of the music

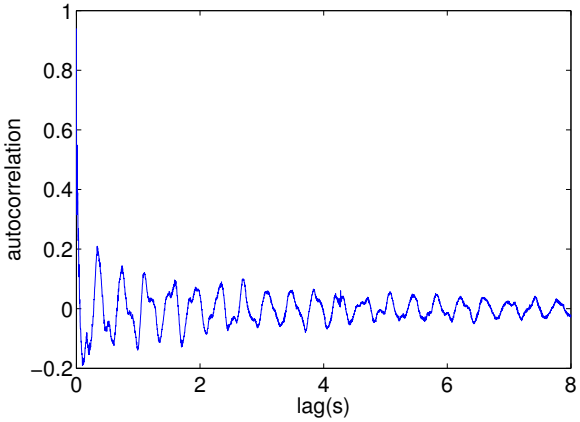


Figure 5: Autocorrelation as function of lag

times. $\alpha \in (0, 1)$ is a weighing parameter to balance the importance of the two terms. $Onset(t_i)$ represents the onset strength of the audio signal at time t_i, while $dist(t_i - t_{i-1}, C)$ represents the difference between the beat interval and the global tempo C, which is determined as described above. We use dynamic programming to optimize the objective function by recursively finding each t_i and hence the best $\boldsymbol{T}$.

5. DANCING PLANNING

In this section, we explain how musical emotions and beat times automatically determines the behaviour of the robot. The musical emotion decides the motion primitive, while beat times control fine timing and synchronization.

5.1 Generate Schedule of Motion Primitives

To generate a schedule of motion primitives according to the emotions of the music, we need to have emotion labels assigned to the motion primitives. It is time-consuming to label each motion primitive with an AV value, so we developed an algorithm that estimates the AV value from emotion labels of the static postures within the motion primitive.

5.1.1 *Mapping Motion Primitive to Activation-Valence Space From Static Postures Data*

We collected 4 static postures of the NAO humanoid robot for each of Ekman's 6 basic emotions: Happy, Sad, Angry, Surprised, Fear and Disgust. Hence, we have a total of 24 emotional static postures. Motion primitives are contructed from these. The arms and head of a NAO humanoid robot can be freely positioned, but there are only 5 different heights and 5 different tilts of the robot to choose from as shown in Figure 6. Static postures to express each emotion using the NAO humanoid robot were collected independently. Figure 7 shows a subset of the data we collected.

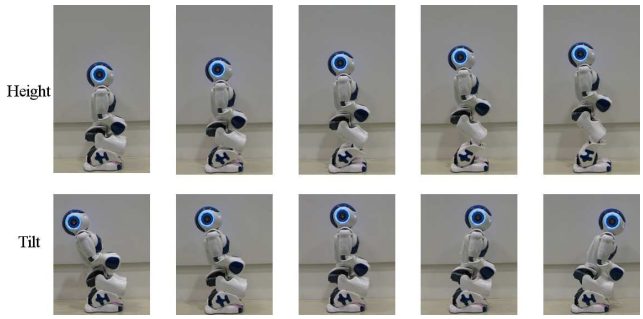


Figure 6: 5 heights and 5 tilts of the NAO robot

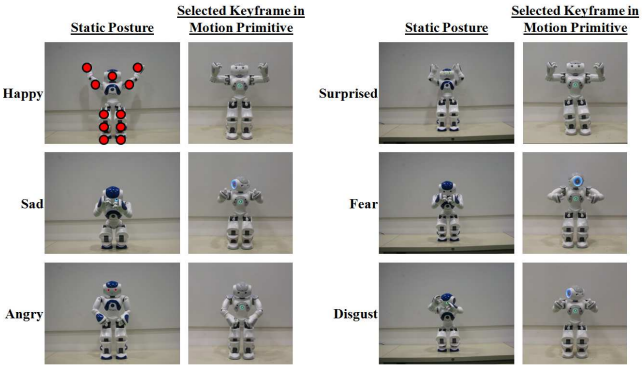


Figure 7: Examples of static postures and corresponding selected keyframes in motion primitives. Red circles indicate the points of interest

Table 1: AV Value for Paul Ekman's six emotions

Emotion	Activation (A)	Valence(V)
Happy	1	1
Sad	-1	-1
Angry	1	-1
Surprised	1	0
Fear	0.5	-1
Disgust	-0.5	-0.5

We assigned each of the 6 basic emotions with an AV value shown in Table 1. We recorded the joint angles of each emotional static posture in a keyframe. To label each motion primitive, we form a weighted sum based on the similarity between motion primitives and static postures with known AV values. SP[em] returns a vector of four static postures (keyframes), where em $\in$ {Happy, Sad, Angry, Disgusted, Fear, Surprised}, e.g., SP[Happy] returns a vector of four keyframes that reflect the Happy emotion. Algorithm 1 first determines the most similar static posture for each *em* to the motion primitive by using the points of interest (POI),

shown in Figure 7. The POI are chosen based on the concepts of markers in motion capture systems. Algorithm 1 calculates DIST, the sum of the Euclidean distances between the 3-dimensional positions of the POI in each static posture with emotion *em* and the 3D positions of POI of the keyframes in the motion primitive $M_{c,n}$ with the function GetDist and returns the least sum of Euclidean distances. Next, Algorithm 2 ranks the least sum of Euclidean distances from each emotion and computes an exponential weighting for each emotion based on its ranking and the Euclidean distance. Lastly, Algorithm 3 estimates the AV values of the motion primitives with the weights found and the AV values in Table 1. Figure 7 shows keyframes from motion primitives that best reflect the emotions. Figure 8 shows the estimated emotion values of all motion primitives.

Algorithm 1 Calculate the least sum of Euclidean distances of points of interest of a motion primitive $M_{c,n}$ and the emotional static posture in SP[em]

GetLeastDiffEm($M_{c,n}, em$)

for KF = 1 to |SP[em]| **do**
  total ← 0
  for kf = 1 to numOfKeyframes($M_{c,n}$) **do**
    total ← total + GetDist($M_{c,n}$[kf], SP[em][KF])
  end for
  DIST[KF] ← total
end for
return $\min_{\text{KF}}$(DIST[KF])

Algorithm 2 Calculate the vector of weights based on the ranking of the Euclidean distances

GetWeightsBasedRank(distances)

for $i = 1$ to |distances| **do**
  flippedDistances[i] ← ($\sum_j$ distances[j]) − distances[i]
end for
sorted ← sortAscending(flippedDistances)
meanValue ← mean(flippedDistances)
for $i = 1$ to |flippedDistances| **do**
  weights[i] ← $e^k + \frac{\text{flippedDistances}[i]}{\text{meanValue}}$ where sorted[k] = flippedDistances[i]
end for
for $i = 1$ to |weights| **do**
  weights'[i] ← $\frac{\text{weights}[i]}{\sum_j \text{weights}[j]}$
end for
return weights'

5.1.2 The Markov Dancer Model

Suppose there is a dancer who dances with a piece of music. At each time point, the dancer strives both to reflect the emotion in the music and to achieve continuity of motions. We want to generate a schedule of motion primitives by mimicking this process. To be specific, this problem is modeled as a Markov chain, which is a generative stochastic motion model. A separate model (Figure 9) is used for for each category, e.g., Head. A Markov chain is used to select the motion primitives $M_{c,i}$ for each schedule S_c (S_{Head}, ..., S_{Legs}). We want to generate $M_{c,i}$ with the probability

Algorithm 3 Estimate AV value of $M_{c,n}$

GetActivationValence($M_{c,n}$)

for emID = 1 to 6 **do**
  emDiff[emID] ← GetLeastDiffEm($M_{c,n}$, emID)
end for
weights ← GetWeightsBasedRank(emDiff)
act ← 0
val ← 0
for emID = 1 to 6 **do**
  act ← act + weights[emID] ∗ em[emID].activation
  val ← val + weights[emID] ∗ em[emID].valence
end for
act ← bound(act, -1, 1)
val ← bound(val, -1, 1)

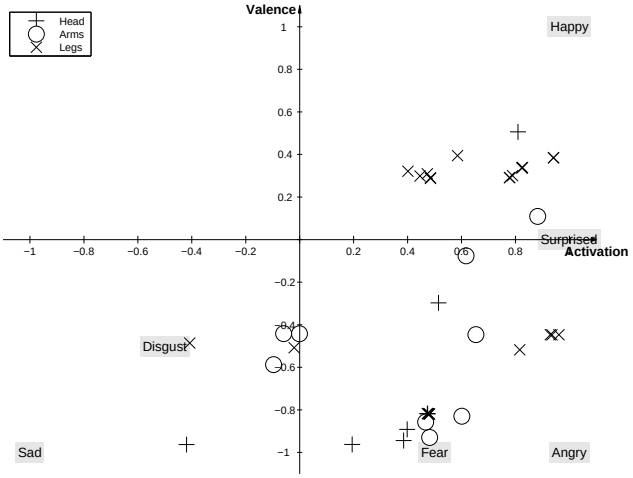


Figure 8: Labeled motion primitives (AV values)

$P(M_{c,i}|M_{c,i-1}, e)$, where e is the emotion detected at the end of $M_{c,i-1}$. As a special case, when $i = 1$, we select $M_{c,1}$ according to $P(M_{c,1}|e)$.

The motion primitive sequence generated by this model should (i) be continuous, (ii) reflect the musical emotion, and (iii) be interestingly non-deterministic. We set the probability function according to Equation 2.

$$P(M_{c,i}|M_{c,i-1}, e) = C \cdot E \cdot N \tag{2}$$

Here, we call C and E the *continuity factor* and *emotion factor*, respectively. They are based on the transition between different motion primitives and the emotion-motion primitive relationships. N is a constant normalizing factor.

Continuity factor: The continuity factor is designed to encourage continuity from each motion primitive to the next. Specifically, we want a quick and smooth interpolation from the last key frame of the current motion primitive to the first key frame of the next motion primitive. We denote the minimum required time interval computed from Algorithm 4 of this interpolation by $dist_M(M_{c,i+1}, M_{c,i})$ in Equation 3.

$$C = \frac{1}{\sqrt{2\pi\sigma_M^2}} \exp\left(-\frac{dist_M^2(M_{c,i+1}, M_{c,i})}{2\sigma_M^2}\right) \tag{3}$$

Here, σ_M^2 is a constant. The continuity factor is big when the minimum transition time is short.

Emotion factor: The emotion foctor is designed to se-

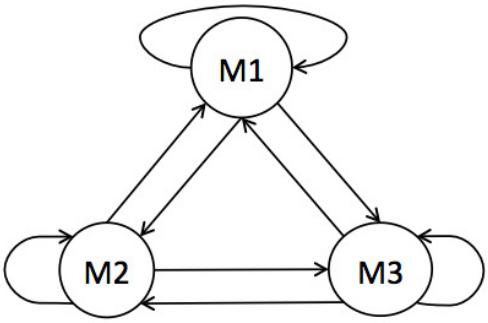


Figure 9: An example of the Markov dancer model shown with only 3 motion primitives for simplicity

lect motion primitives whose emotions are similar to the musical emotion. We denote the (a, v) coordinate of $M_{c,i}$ by $E(M_{c,i})$, and denote the Cartesian distance between $E(M_{c,i})$ and e on the AV plane by $dist_e(E(M_{c,i}), e)$ in Equation 4.

$$E = \frac{1}{\sqrt{2\pi\sigma_e^2}} \exp\left(-\frac{dist_e^2(E(M_{c,i}), e)}{2\sigma_e^2}\right) \quad (4)$$

Here, σ_e^2 is a constant. The emotion factor is big when the emotional difference is small. Again, e refers to the detected emotion at the end of $M_{c,i-1}$.

5.2 Determine Interpolation Times and Timing Parameters β in Schedule of Motion Primitives

After describing the process to select the sequence of motion primitives, we provide an algorithm to synchronize the schedule of motion primitives with the detected beat times, where each motion primitive in the schedule should end on a beat time. When a motion primitive ends, we begin interpolating to the first keyframe of the next motion primitive.

5.2.1 Calculate Time to Interpolate Between Motion Primitives

Algorithm 4 calculates the time needed to interpolate from the last keyframe, K_j, of the previous motion primitive $M_{c,m-1}$, to the first keyframe, K_l, of $M_{c,m}$, using the joint angles V_j of K_j and V_l of K_l. Although we can interpolate between two keyframes with maximum joint angular speeds given the joint angles, we want the robot to dance stably. As we do not implement the controller for the actuators of the robot to account for dynamics, we weight the minimum duration for the interpolation with a multiplier in Algorithm 4. We define λ as the maximum time multiplier, where $\lambda = 0.4$ ($e^{0.4} \approx 1.5$) so that the maximum time multiplier $\leq 1.5\gamma$. We define γ for each category (Table 2). E.g., we assign a higher γ of 3 for the legs and 1.5 for the head, so that the robot's legs move slower than the head and the robot is more stable at the bottom. We weighted the time multiplier more heavily when the $avgtime \approx maxtime$ which implies that all the joints move almost equally fast.

5.2.2 Calculate Timing Parameter β

The time required for each motion primitive includes the interpolation time between two primitives computed from Algorithm 4 and the times between the keyframes in the motion primitive. If only one beat-time interval is insufficient to execute the motion primitive, we add subsequent beat-time intervals until the total time offered is long enough

Algorithm 4 Calculate duration required to interpolate from K_j to K_l

GetDuration(K_j, K_l)
1: **for** JI = 1 to $|K_j|$ **do**
2: time[JI] $\leftarrow |V_{l,\text{JI}} - V_{j,\text{JI}}| \div \text{MaxAngularSpeed}[JI]$
3: **end for**
4: maxTime $\leftarrow$ max(time)
5: avgTime $\leftarrow$ average(time)
6: **if** maxTime = 0 **then**
7: **return** 0
8: **end if**
9: timeMultiplier $\leftarrow e^{\text{avgTime/maxTime}*\lambda} * \gamma$
10: **return** maxTime $*$ timeMultiplier

Table 2: γ values for joint categories

Category	Head	Arm	Leg
γ	1.5	2	3

for execution (Figure 10). To make each motion primitive end at a beat time, we stretch the duration by increasing the parameter, β, in each motion primitive to fill the time interval from its starting beat time to the next beat time. In practice, the schedule of motion primitives for each body part is planned independently and executed simultaneously.

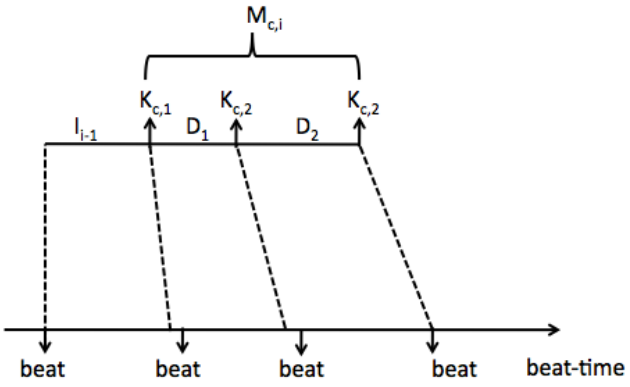


Figure 10: Synchronizing motion primitive with beat times

5.2.3 Emotion For Next Motion Primitive

Motion primitives are selected sequentially and stretched to fill a whole number of beat times. To choose the next motion primitive, we need the emotion at the end of the previous motion primitive. We simply estimate the emotion at each beat time by linearly interpolating the (a, v) values, which are computed at 15-second intervals.

6. EXECUTION

The schedule of motion primitives computed in Section 5.1 needs to be synchronized to the music based on beat times during execution. Since there are always latency and other differences between desired timing and real timing in robot motion, the starting and ending times of planned motion primitives will diverge from the plan and therefore become totally out of phase with respect to the music. To correct this drift, we use an adaptive real-time synchronizing algorithm (Algorithm 5), which is inspired by work on real-time

automatic music accompaniment [12]. Here, *start* and *end* are the two variables keeping track of ideal starting and ending times of each motion primitive $M_{c,m}$. Due to the execution error, the motion $M_{c,m-1}$ will not precisely end at its ideal ending time. Therefore, $M_{c,m}$ will start whenever $M_{c,m-1}$ ends and we recalculate the *duration* from the actual real time returned by the function, getTime. The duration is calculated so that $M_{c,m}$ will end at the ideal end time. Again, $M_{c,m}$ will not actually finish at exactly this ideal ending time. Therefore, $M_{c,m+1}$ will again calculate a *duration* and move on. This algorithm adjusts the execution duration of every motion primitive by updating the parameter, β, in the motion primitive using the function updateBeta to avoid the accumulated timing errors. updateBeta adjusts the total time that it originally takes to execute the motion primitive to be the same as the updated *duration* by changing β of the motion primitive.

Algorithm 5 Adaptive Real-time Synchronizer

Synchronizer(S_c)
1: *start* ← getTime()
2: **for all** $M_{c,m}$ in S_c **do**
3: *end* ← *start* + $M_{c,m}$.duration
4: *duration* ← *end* − getTime()
5: updateBeta($M_{c,m}$, *duration*)
6: execute($M_{c,m}$)
7: *start* ← *end*
8: **end for**

7. RESULTS

Table 3 is a contrast experiment to show how the continuity and emotion factors affect the plan for a Pleased piece of music and right arm motion primitives as an example. The first column is the average minimum duration for the interpolation from one motion primitive to the next one. Smaller numbers indicate greater continuity. The second column is the average Euclidean distance between the emotion of the motion primitives and the emotion of the music on the AV plane. Smaller numbers indicate greater correspondence between the dance emotion and the music emotion. The first row is the experimental trial, which takes both the continuity and emotion factors into account, while the second and the third row are control trials, which eliminate the continuity factor and emotion factor, respectively. The fourth row is also a control trial, which eliminates both factors and generates a random dancing plan. The results show that both emotion and continuity factors are beneficial.

Table 3: A contrast experiment to show how continuity and emotion affect dancing plan

Trial	Behavior distance	Emotion distance
E and C	0.55	0.61
E	0.94	0.49
C	0.4	0.63
R	0.79	0.68

Figure 11 shows a planned schedule of motion primitives for the *RArm* for a short snippet of Peaceful music whereas Figure 12 shows a planned schedule of motion primitives the *RArm* for Angry music. Figure 11 and Figure 12 plot the music signals and beat times and we show that the planned schedule of motion primitives corresponds to the beats.

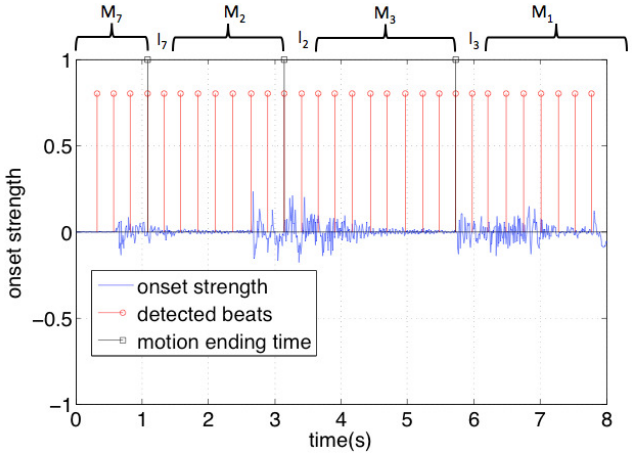


Figure 11: RArm motion primitives schedule for Peaceful music

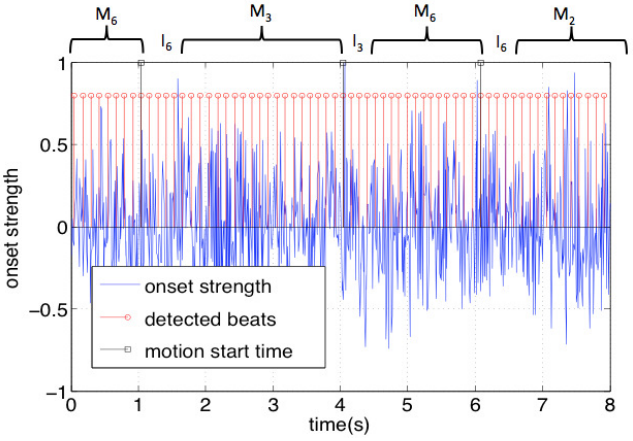


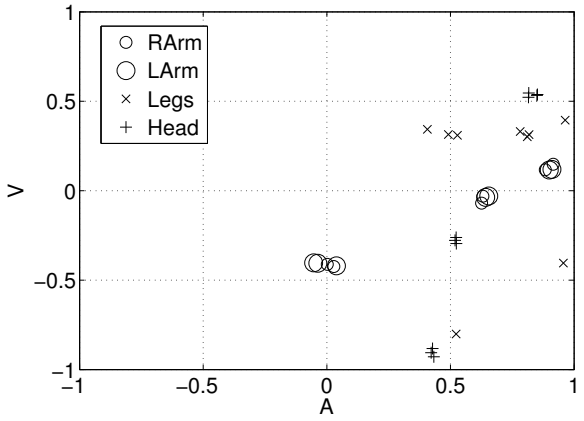
Figure 12: RArm motion primitives schedule for Angry music

We also show that different motion primitives are selected for Pleased music and Angry music in Figure 13. Figure 14 shows how the real-time synchronizer reduces timing errors. The dotted line shows executed time using real-time synchronizer and is a good match to the ideal time. The black line shows the executed time without a real-time synchronizer, which is totally out of phase after a minute.

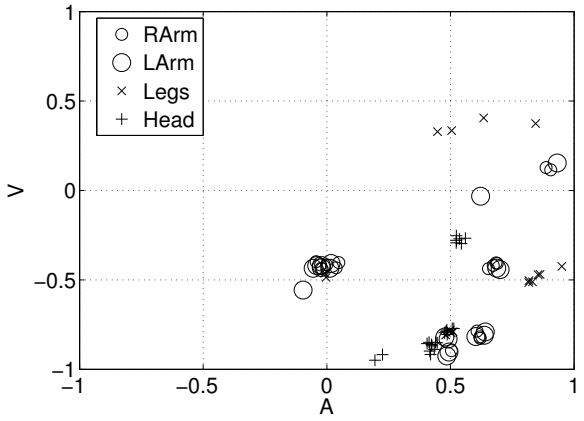
Figure 15 shows snapshots of the NAO humanoid robot dancing with a piece of Angry music. Most of these snapshots show the NAO robot leaning forward, the head bent forward, and putting the arms at the side of the body. These postures are similar to the Angry static postures collected.

8. CONCLUSIONS

We show that we can automate robot dancing by forming schedules of motion primitives that are driven by the emotions and the beats of any music on a NAO humanoid robot. The algorithms are general and can be used on any robot. From emotion labels given for static postures, we can estimate the activation-valence space locations of the motion primitives and select the appropriate motion primitives for emotions detected in music. We also show that we can monitor the execution of the schedule of motion primitives and compensate for any timing errors found, ensuring synchro-



(a)



(b)

Figure 13: Activation-Valence coordinates of motion primitives chosen for (a) Pleased music, and (b) Angry music. Motion primitives visited multiple times are drawn with slight offsets to convey their number.

nization between robot dance motions and the music.

Acknowledgments

This work is supported by a generous gift awarded to the School of Computer Science, Carnegie Mellon University. We wish to thank Byeong-jun Han for the comments on music emotion recognition. We also wish to thank Somchaya Liemhetcharat for his help on data collection and feedback on the algorithms.

9. REFERENCES

[1] S. Bock and M. Schedl. Enhanced beat tracking with context-aware neural networks. In *Proc. Int. Conf. Digital Audio Effects*, 2011.

[2] C. Breazeal. Emotion and sociable humanoid robots. *Int. J. of Human-Computer Studies*, 59:119–155, 2003.

[3] P. Ekman. Are there basic emotions? *Psychological Review*, 99(3):550–553, 1992.

[4] D. Ellis. Beat tracking with dynamic programming. *MIREX Audio Beat Tracking Contest sys. desc.*, 2006.

[5] A. J. Eronen and A. P. Klapuri. Music tempo estimation with k-nn regression. *Trans. Audio, Speech and Lang. Proc.*, 18(1):50–57, 2010.

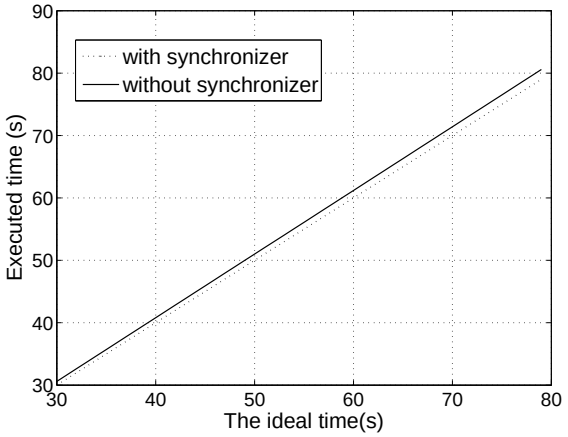


Figure 14: Executed time difference: with a real-time synchronizer (RTS) vs. no RTS

Figure 15: Video keyframes of NAO humanoid robot dancing with Angry music

[6] M. Goto. A study of real-time beat tracking for musical audio signals. *Ph.D. thesis*, 1998.

[7] B. Han, S. Rho, R. Dannenberg, and E. Hwang. SMERS: Music emotion recognition using support vector regression. In *ISMIR'09*, pages 651–656, 2009.

[8] G. Kim, Y. Wang, and H. Seo. Motion control of a dancing character with music. In *IEEE/ACIS Int. Conf. Comp. Info. Science*, pages 930–936, 2007.

[9] T.-h. Kim, S. Park, and S. S. Y. Rhythmic-motion synthesis based on motion-beat analysis. *ACM Transactions on Graphics*, 2003.

[10] R. Kirby, R. Simmons, and J. Forlizzi. Modeling affect in socially interactive robots. In *Proc. Int. Symp. Robot Human Interact. Comm.*, pages 558–563, 2006.

[11] H. Lee and I. Lee. Automatic synchronization of background music and motion. *Computer Graphics Forum*, 24:353–362, 2005.

[12] D. Liang, G. Xia, and R. Dannenberg. A framework for coordination and synchronization of media. In *Proc. Int. Conf. New Interfaces Musical Expr.*, 2011.

[13] S. Nakaoka, S. Kajita, and K. Yokoi. Intuitive and flexible user interface for creating whole body motions of biped humanoid robots. In *IEEE Int. Conf. Intelligent Robots and Systems*, pages 1675–1682, 2010.

[14] J. Oliveira, L. Naveda, F. Gouyon, M. Leman, and L. Reis. Synthesis of variable dancing styles based on a compact spatiotemporal representation of dance. In *IEEE Int. Conf. Intelligent Robots and Systems*, 2010.

[15] T. Shiratori, A. Nakazawa, and K. Ikeuchi. Dancing-to-music character animation. *Computer Graphics Forum*, 25:449–458, 2006.

[16] R. E. Thayer. *The Biopsychology of Mood and Arousal*. Oxford University Press, New York, 1989.

Session 1B
Teamwork I

Coordination Guided Reinforcement Learning

Qiangfeng Peter Lau[♠], Mong Li Lee[†] and Wynne Hsu[§]
Department of Computer Science
National University of Singapore
13 Computing Drive, Singapore 117417, Republic of Singapore
{plau[♠],leeml[†],whsu[§]}@comp.nus.edu.sg

ABSTRACT

In this paper, we propose to guide reinforcement learning (RL) with expert coordination knowledge for multi-agent problems managed by a central controller. The aim is to learn to use expert coordination knowledge to restrict the joint action space and to direct exploration towards more promising states, thereby improving the overall learning rate. We model such coordination knowledge as constraints and propose a two-level RL system that utilizes these constraints for online applications. Our declarative approach towards specifying coordination in multi-agent learning allows knowledge sharing between constraints and features (basis functions) for function approximation. Results on a soccer game and a tactical real-time strategy game show that coordination constraints improve the learning rate compared to using only unary constraints. The two-level RL system also outperforms existing single-level approach that utilizes joint action selection via coordination graphs.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, Search

General Terms

Algorithms, Performance, Experimentation

Keywords

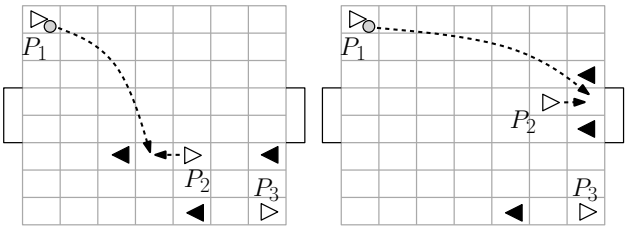
Reinforcement learning, guiding exploration, coordination constraints, factored Markov decision process

1. INTRODUCTION

Expert knowledge is commonly employed in large-scale reinforcement learning (RL) in a variety of ways. In particular, hierarchical RL handles single agent Markov decision processes (MDPs) by recursively partitioning them into smaller problems using a task hierarchy [19, 7, 1]. The task hierarchy constrains the solution space (policies) of the learning problem so that only relevant actions for a task can be selected at each time step. Learning a good task selection policy will direct exploration towards the more promising parts of the MDP.

For multi-agent problems, each agent has a set of actions whose Cartesian product forms the joint action space. This space is exponential in the number of agents and therefore, RL with naive exploration is slow. Hierarchical RL has been adapted to multi-agent problems [15, 9] by having one task hierarchy per agent where the actions are selected jointly. Once each individual agent's task is selected, it will have a constrained (reduced) set of actions to consider. However, this framework cannot be easily extended to incorporate coordination behavior among multiple agents.

Consider Fig. 1 which depicts a state in a soccer game and player P_1 has the ball. Let N, S, E, W, be the four compass directions. P_1's action set is $A_1 = \{S, E, pass2, pass3, shoot\}$ where $pass2$ and $pass3$ denote passing the ball to players P_2 and P_3 respectively, and $shoot$ denotes the action to kick the ball into the goal. Players P_2 and P_3 have the action set $A_2 = \{N, S, E, W\}$ and $A_3 = \{N, W\}$ respectively. We denote a joint action as $\langle a_1, a_2, a_3\rangle \in A_1 \times A_2 \times A_3$. The size of this joint action space is $5 \times 4 \times 2 = 40$. A closer examination reveals that much of this space does not need be explored as they are unlikely to lead to a winning state. For example, P_1 certainly should not pass the ball to P_2 if P_2 is moving adjacent to an opponent as the ball can easily be intercepted. With this simple coordination strategy, the set of disallowed joint actions is $\{pass2\} \times \{S, E, W\} \times A_3$. Similarly, P_1 should not pass the ball to P_3 and the set of disallowed joint actions is $\{pass3\} \times A_2 \times A_3$. Immediately, the size of the joint action space is reduced by 35%.



(a) Bad pass (b) Good pass

Figure 1: Example states in a simplified soccer game. The white ▷ versus black ◀ players.

In this paper, we focus on a central learner with multiple agents where communication is free. This corresponds to the scenario of a computer player managing an army in real-time strategy (RTS) games or a team of players in soccer. We aim to exploit coordination knowledge for improving the learning rate of good policies by modeling coordination among agents

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

as hard constraints. We refer to these hard constraints as coordination constraints (CCs), and use CCs to limit the joint action space for exploration. Unary constraints defined on single agents are a special case of CCs.

We propose a two-level RL system where the top level learns to choose the CCs to constrain the bottom level's learning of joint actions. The RL system learns to guide itself, i.e., deciding which CCs to use in various states is part of the learning process. This is a necessary flexibility because not all CCs are always useful in every state. For example, in Fig. 1a, the CC that P_1 should not pass the ball to P_2 is appropriate since there are opponents close to P_2. However, this CC may not be suitable, e.g., in Fig. 1b, where P_2 is standing directly in front of the goal as it may be better to pass to P_2 so that P_2 can try to score. Such a two-level system is different from RL for constrained MDPs [8] as the two-level system dynamically learns to use different constraints to improve learning, instead of using static constraints to prevent failure.

The proposed CCs are useful in addition to existing methods of modeling coordination with a communication structure such as a coordination graph (CG) for joint action selection [11, 12]. Unlike task-based methods where single agents are restricted based on individual tasks, CCs define these restrictions based on expert knowledge of multi-agent coordination. Such coordination is not easily expressed within single agent tasks. Existing works usually delegate them to the CG structure as features (basis functions) [15] or as static rules [16]. This further motivates us to investigate the potential of CCs in a more active role for improving learning performance.

Using CCs for online RL has three challenges. First, the system must be able to efficiently learn to activate the various CCs from its interaction with the environment. However, different combinations of activated CCs lead to large number of bottom level value functions to be learned. We address this by formulating learning equations that exploit similarities among the bottom level components of our system. Second, choosing CCs at the top level introduces an exponential top level joint action space to explore. We overcome this by identifying those CCs which can never be violated in a given state. Once identified, these CCs are deactivated, reducing the top level action space for exploration. Last, the system must integrate well with other useful methods for multi-agent learning in large state-action spaces.

To the best of our knowledge, this is the first online RL system that uses coordination to guide learning in multi-agent problems with a central controller. Our model-free approach frees the user from designing models for large MDPs with many variables. A major benefit of our system is that existing predicate definitions of features can be reused to specify CCs. This sharing of predicate components between CCs and features aids the user in encoding knowledge for the top level of the system. Experiments show that CCs give better results compared to having unary constraints or with coordination knowledge encoded only as a CG. For domains that require heavy coordination, using CCs leads to better policies, and hence better overall goal achievement.

2. TWO-LEVEL RL SYSTEM

An MDP is a 4-tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of primitive actions, $\mathcal{P}(s'|s,a)$ is a transition probability model that gives the probability of going from state s to s' when action a is taken, and $\mathcal{R}(s,a,s')$ is a reward model that gives the reward of taking a in s and reaching s'. The set of available actions at state s is written as $\mathcal{A}(s)$. With N agents, the joint action space is factored as $\mathcal{A} = A_1 \times ... \times A_N$, where A_n is the action variable that corresponds to the n-th agent. $\mathcal{S}$ may also be factored into multiple variables in a similar way.

A solution to the MDP is a policy $\pi : \mathcal{S} \mapsto \mathcal{A}$ and the optimal policy π^* is one that maximizes the expected total discounted reward in any given state. Let the expected discounted sum of rewards when taking a in s at time t, observing reward r_t, and following π thereafter with discount rate γ be $Q^\pi(s,a) = E_\pi\{\sum_{t'=t}^{\infty} \gamma^{t'-t} r_{t'} | s, a\}$. Then $\pi^*(s) = \text{argmax}_{a \in \mathcal{A}(s)} Q^*(s,a)$. By learning Q^* directly, we obtain π^* without learning $\mathcal{P}$ and $\mathcal{R}$ [18].

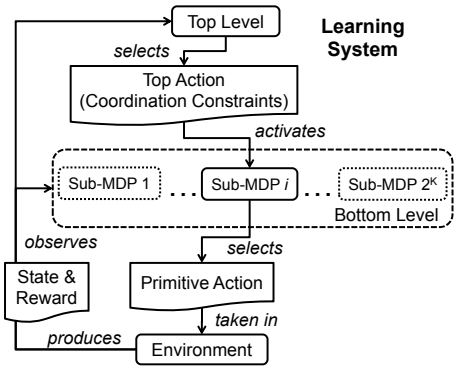


Figure 2: Two-level learning system

Fig. 2 depicts the proposed two-level learning system. The top level determines which CCs are relevant to the current state. The top level action space $\mathcal{A}_0$ consists of the *activation* or *deactivation* of each CC. With K constraints, the size of $\mathcal{A}_0$ is 2^K. However, in practice, the number of CCs that may be activated is typically small. The activated CCs restrict the bottom level to a sub-MDP, $i \in [1, 2^K]$, as shown in Fig. 2. Therefore, sub-MDP i corresponds to a unique top level action in $\mathcal{A}_0$. The joint action space of sub-MDP i, denoted by $\mathcal{A}_i$, is a subset of the original joint action space $\mathcal{A}$. This allows the bottom level to learn the original joint actions quickly. After the bottom level has taken an action in the environment, execution returns to the top level. Details of the learning equations, action selection and specification of CCs are given in Sections 2.1, 2.2 and 2.3. We also discuss how $\mathcal{A}_0$ can be reduced in Section 2.4.

2.1 Learning Equations & Updates

To model the system's two-level learning, we augment the original MDP's state space with an index i that keeps track of the position within the hierarchy. The top level corresponds to $i = 0$, and $i \in [1, 2^K]$ refers to one of the sub-MDPs. Note that when $i \in [1, 2^K]$, it also refers to a unique top level action in $\mathcal{A}_0$. The *augmented* MDP's state space is $\mathcal{S}' = [0, 2^K] \times \mathcal{S}$, and $\langle i, s \rangle \in \mathcal{S}'$ is an augmented state. The augmented action space is the union $\mathcal{A}' = \mathcal{A}_0 \cup \mathcal{A}$. Let an action in $\mathcal{A}'$ be $\tilde{a}$. Then, the transitions between levels in the hierarchy are deterministic while those between original states follow $\mathcal{P}$, resulting in the transition model,

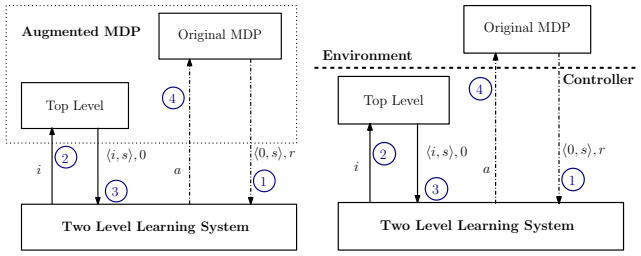
$$\mathcal{P}'(\langle i', s' \rangle | \langle i, s \rangle, \tilde{a}) = \begin{cases} \mathcal{P}(s'|s,\tilde{a}) & \text{if } i \neq 0 \wedge i' = 0 \wedge \tilde{a} \in \mathcal{A} \\ 1 & \text{if } i = 0 \wedge i' = \tilde{a} \wedge \tilde{a} \in \mathcal{A}_0 \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

and the reward model is

$$\mathcal{R}'(\langle i,s\rangle,\tilde{a},\langle i',s'\rangle)=\begin{cases}\mathcal{R}(s,\tilde{a},s') & \text{if } i'=0\\ 0 & \text{otherwise}\end{cases} \quad (2)$$

that is, the original reward model $\mathcal{R}$ is used when transiting between original states, all other transitions are zero. Note that the augmented MDP is also an MDP.

Fig. 3 illustrates the dynamics of the augmented MDP from the system's point of view and that of the original environment. Solid arrows indicate deterministic transitions while dotted-dashed arrows indicate non-determinism. Numbers describe a sequence of transitions between two original states. In Fig. 3a, the RL system operates as if the top level is part of the environment. Conversely in Fig. 3b, the original environment only receives original joint actions from the central controller that consists of multiple agents.



(a) RL System View (b) Environment View

Figure 3: Interactions between RL system, augmented MDP, and environment (MDP).

Let the hierarchical policy, $\pi:\mathcal{S}'\mapsto\mathcal{A}'$, that solves the augmented MDP be represented by a set of policies $\{\pi_i\}$ indexed by i such that $\pi(\langle i,s\rangle)=\pi_i(s)$. That is, π_0 denotes the top level policy, and $\pi_{i>0}$ denotes a policy constrained to the action space $\mathcal{A}_i(s)$ of the sub-MDP i. The Bellman equation of the action value function for the augmented MDP is,

$$Q^\pi(\langle i,s\rangle,\tilde{a})=\sum_{\langle i',s'\rangle\in\mathcal{S}'}\mathcal{P}'(\langle i',s'\rangle|\langle i,s\rangle,\tilde{a}) \times[\mathcal{R}'(\langle i,s\rangle,\tilde{a},\langle i',s'\rangle)+Q^\pi(\langle i',s'\rangle,\pi_i(s'))]. \quad (3)$$

Suppose we only use discounting (γ) for transitions between original states, we can rewrite and simplify Eq. 3 into two parts, $\forall i>0$, for the top level,

$$Q^\pi(\langle 0,s\rangle,i)=\sum_{s'\in S}\mathcal{P}_i(s'|s,\pi_i(s))[\mathcal{R}(s,\pi_i(s),s') +\gamma Q^\pi(\langle 0,s'\rangle,\pi_0(s'))] \quad (4)$$

and for the bottom level,

$$Q^\pi(\langle i,s\rangle,a)=\sum_{s'\in S}\mathcal{P}_i(s'|s,a)[\mathcal{R}(s,a,s') +\gamma Q^\pi(\langle 0,s'\rangle,\pi_0(s'))]. \quad (5)$$

Note that i in Eq. 4 refers to a unique top level action in $\mathcal{A}_0$, and we subscript $\mathcal{P}$ with i to indicate primitive actions that are disallowed based on i. Eq. 4 expresses the expected reward of taking a top level action i and following π_i for one step before returning to the top and following π_0 thereafter. Eq. 5 expresses the expected reward at the bottom level that returns to the top level immediately after one step and following π_0 thereafter. Now we can select greedy primitive actions in s by separately computing $\pi_0(s)=\text{argmax}_{i\in\mathcal{A}_0(s)}Q^\pi(\langle 0,s\rangle,i)$, followed by $\pi_i(s)=\text{argmax}_{a\in\mathcal{A}_i(s)}Q^\pi(\langle i,s\rangle,a)$.

Updating the 2^K $Q^\pi(\langle i,\cdot\rangle,\cdot)$ functions at the bottom level requires exponential space and time. However we observe that, although $\mathcal{P}_i$ and $\mathcal{P}_j$ for two sub-MDPs have different domains as their actions are from $\mathcal{A}_i$ and $\mathcal{A}_j$ respectively, their probabilities are contained in the original MDP's $\mathcal{P}$. This is because the transition probability is a conditional probability where values are normalized over the state space but not the action space. Hence, given 2^K sub-MDPs, $\forall s\in\mathcal{S},i,j\in[1,2^K],a\in\mathcal{A}_i(s)\cap\mathcal{A}_j(s)$, we have

$$Q^\pi(\langle i,s\rangle,a)=Q^\pi(\langle j,s\rangle,a). \quad (6)$$

Consequently, the various sub-MDP functions are the same for their intersected domains. This leads to a single bottom level function definition: $\forall i>0,a\in\mathcal{A}_i(s)$,

$$U^\pi(s,a)=Q^\pi(\langle i,s\rangle,a) \quad (7)$$

that is independent of i. Incidentally, U^π is similar to the action value function for the original MDP but constrained to the joint actions in $\mathcal{A}_i(s)$ by the two-level policy π.

We use linear function approximation to learn the value functions $Q^\pi(\langle 0,\cdot\rangle,\cdot)$ and U^π. This employs a linear combination of m number of features (basis functions), ϕ_p, with weights, w_p, to be learned. In other words, given a function $F(s,a)$, we want to find $\vec{w}=\langle w_1,...,w_m\rangle$ such that, $F(s,a)\approx\vec{w}\cdot\vec{\phi}_{s,a}$, where $\vec{\phi}_{s,a}=\langle\phi_1(s,a),\cdots,\phi_m(s,a)\rangle$.

To obtain $\vec{w}$ for each optimal value function, we perform on-policy temporal difference (TD) updates using the hierarchical policy π given by $\{\pi_i\}$, where each π_i is a GLIE policy [7], and online samples of the form $\langle\langle 0,s\rangle,i,\langle i,s\rangle,a,r,\langle 0,s'\rangle\rangle$. The first two entries in the sample denote that the top level is in the augmented state $\langle 0,s\rangle$ and it chooses the action $\pi_0(s)=i\in\mathcal{A}_0(s)$. The next two entries in the sample indicate the state $\langle i,s\rangle$ of the bottom level and the primitive action $\pi_i(s)=a\in\mathcal{A}_i(s)$ taken by it. The final two entries indicate that both levels observe reward r and go to next state s'. Note that when the bottom level policy chooses an exploratory action, i.e., $\pi_i(s)$, it does so by choosing a random action within the constrained joint action space $\mathcal{A}_i(s)$ as specified by the top level action i.

Let $Q^\pi(\langle 0,s\rangle,i)\approx\vec{w_0}\cdot\vec{\phi_0}_{s,i}$, $U^\pi(s,a)\approx\vec{w_U}\cdot\vec{\phi_U}_{s,a}$, and α be the step size parameter that decreases over time. Then, the weights $\vec{w_0}$ and $\vec{w_U}$ are updated as follows:

$$\vec{w_0}\leftarrow\vec{w_0}+\alpha[r+\gamma Q^\pi(\langle 0,s'\rangle,\pi_0(s')) -Q^\pi(\langle 0,s\rangle,i)]\vec{\phi_0}_{s,i} \quad (8)$$

$$\vec{w_U}\leftarrow\vec{w_U}+\alpha[r+\gamma Q^\pi(\langle 0,s'\rangle,\pi_0(s')) -U^\pi(s,a)]\vec{\phi_U}_{s,a} \quad (9)$$

2.2 Action Selection

Applying the policies π_i often requires selecting some primitive action within $\mathcal{A}_i(s)$ that maximizes the value functions. For example, the ϵ-greedy bottom level policy is to select a maximal action $\pi_i(s)=\text{argmax}_{a\in\mathcal{A}_i(s)}U^\pi(s,a)$ with $1-\epsilon$ probability, or a random action within $\mathcal{A}_i(s)$ with ϵ probability. In our system, $\mathcal{A}_i(s)$ is subjected to the constraints activated by the top level. This implies that the problem

of finding a maximal action, $\text{argmax}_{a \in \mathcal{A}_i(s)} U^\pi(s,a)$, can be modeled as a constraint optimization problem (COP) over the full original action space $\mathcal{A}$ as follows:

$$\underset{a \in \mathcal{A}}{\text{argmax}}\, U^\pi(s,a), \qquad \text{subject to:} \qquad (10)$$
$$c_{i,1}(s,a), ..., c_{i,p}(s,a) \quad c_{0,1}(s,a), ..., c_{0,q}(s,a)$$

where the constraints $c_{i,j}(s,a)$ are activated by the top level action i, termed dynamic CCs, and the constraints $c_{0,j}(s,a)$ are always activated regardless of i, termed static CCs.

Let each constraint be a function on a subset of variables in $\mathcal{S}$ and $\mathcal{A}$ that returns $-\infty$ if violated, or 0 otherwise. Then the objective function to maximize for the COP is

$$U^\pi(s,a) + C_0(s,a) + C_i(s,a) \qquad (11)$$

where C_0 and C_i are the sum of their respective constraints $c_{0,j}$ and $c_{i,j}$. Note that to switch to selecting random actions within $\mathcal{A}_i(s)$, we can simply replace U^π with a random function. Likewise, the selection of action for the top level, e.g., $\text{argmax}_{i \in \mathcal{A}_0(s)} Q^\pi(\langle 0, s\rangle, i)$, can be similarly modeled.

Depending on the characteristics of the COPs, we employ different strategies to solve them. If the problem consists of features and constraints that can be additively decomposed into component functions involving up to two action variables, we can utilize the coordination graph (CG) [11] with bucket elimination for sparse CGs, or the Max-plus algorithm [14] for dense CGs. CGs are formed by having one vertex for each agent (action variable) and an edge between two agents that have a non-zero component function involving them. We further employ domain reduction techniques to prune $\mathcal{A}$. If the problem involves higher arity features and constraints, more generalized solvers can be employed [17]. Furthermore, if top level actions activating CCs are presumed to be independent, we may use features for $Q^\pi(\langle 0, \cdot\rangle, \cdot)$ that only involve the state and one action variable corresponding to one CC. Consequently, top level actions can be selected independently in $O(K)$ time while bottom level actions are selected jointly. This turns out to be sufficient for good performance shown in Section 3.

2.3 Features & Constraints

Next, we show how existing predicate definitions of features can be reused to specify the CCs. We further describe how the top and bottom levels' BFs may share predicate components in their design and highlight a type of features that may be useful for certain multi-agent problems.

Predicates are a natural way to encode expert knowledge as features for RL. For example, the expert knowledge of a bad pass can be written as the predicate:

$$BadPass(s, a_x, a_y) := HasBall(P_x) \wedge IsPass(P_y, a_x) \wedge MoveNextToOpp(s, a_y),$$

where $HasBall(P_x)$ is true if player P_x has the ball, $IsPass(P_y, a_x)$ is true if the action of player P_x, a_x, is to pass the ball to P_y, and $MoveNextToOpp(s, a_y)$ is true if the action of player P_y is to move next to an opponent.

With this predicate $BadPass$, we can derive the corresponding list of propositional features (PFs) by binding the variables P_x, P_y to specific players. For example, for P_1, P_2 we have the PF, $\phi_{BadPass_{1,2}}(s,a) = BadPass(s, a_1, a_2)$, for brevity we write $BadPass_{1,2}$. The value of a PF is in $\{0, 1\}$.

PFs are commonly employed in existing RL systems to approximate the value function [15, 13, 2]. We also utilize PFs for the bottom level function U^π. An immediate advantage is that the predicates for PFs can be reused for specifying CCs. For each PF $\phi_\rho(s,a)$, we can formulate it into a constraint that disallows the proposition ρ, i.e.:

$$c_\rho(s,a) = -\infty \cdot \phi_\rho(s,a). \qquad (12)$$

If $\phi_\rho(s,a) = 1$, $c_\rho(s,a)$ returns $-\infty$, signifying that the constraint c_ρ that represents the condition $\neg\rho$ is violated.

Reusing PFs in U^π as CCs has an added advantage during bottom level action selection. Instead of specifying individual constraints $c_{i,j}$ to sum for C_i in the objective function in Eq. 11, we can simply set the corresponding PFs' weights of the activated CCs to $-\infty$. In so doing, the set of actions that are disallowed by the CCs will never be chosen due to its $-\infty$ weight.

Note that we restore the original weights of these PFs during the updates (Eq. 9). This is because in practice, incomplete COP solvers like Max-plus may still select actions that violate certain activated CCs. When this happens, we can learn the weights for the violated constraint PFs that are useful for updating other weights. Hence PFs can be used both as constraints for guiding exploration and for function approximation.

The top level value function $Q^\pi(\langle 0, \cdot\rangle, \cdot)$ is also a linear approximation. Here, we describe how the predicates for bottom level PFs can be reused for the top level features. We observe that the activation of a constraint is often dependent on the state of the environment. Hence, we encode such state-dependent activation knowledge as the top-level PFs in the following manner: Let $Activated(c)$ be true if constraint c is activated. For each c corresponding to some PF for the bottom level, we conjunct $Activated(c)$ or its negation with selected state predicates of agents involved in c.

For example, in the soccer scenario, we would like to deactivate the $BadPass_{1,2}$ constraint if the receiving player P_2 is near the enemy goal, i.e., $NearGoal(P_2)$ is true as shown in Fig. 1b. This is because it may turn out to be better to take a chance at scoring. Hence, we define a predicate $NearGoal(P_y) \wedge \neg Activated(BadPass_{x,y})$ for each pair of players to capture this condition in the top level value function. This simple strategy allows us to design top level features easily by reusing bottom level features' predicates.

For applications where the agents are homogeneous or their quantity changes over time, a new class of features called relational features (RFs) can be utilized [20]. Here, we modify the relations used in [10, 2] for function approximation by aggregating PFs that share the same predicate into RFs. RFs are additively decomposable into components involving a subset of agents, e.g. an RF of $BadPass_{x,y}$ can be the count of true bound predicates for each pair of players, P_x, P_y. A weight is learned for the RF instead of one for each of the PFs, i.e., the update by observing the true proposition $BadPass_{1,2}$ will have its effect generalized to other pairs of players for $BadPass$. RFs can greatly reduce the number of weights for the top level PFs relating to multi-agent CCs. For example, the predicate $NearGoal(P_y) \wedge \neg Activated(BadPass_{x,y})$ can be used to construct an RF to reduce $N(N-1)/2$ weights to one weight.

2.4 Top Level Learning Efficiency

Finally, we discuss how the 2^K top level action space can be reduced for exploration and consequently, faster learning. In many domains, we observe that the top level action space,

$\mathcal{A}_0$, may be heavily constrained based on the current state, s. This yields a smaller $\mathcal{A}_0(s)$ to explore. In fact, this reduction to $\mathcal{A}_0(s)$ can be directly derived from the predicates used to create the CCs. If it can be inferred that a CC cannot be violated in the current state s, then the CC need not be activated. This can be done in $O(K)$ time as we only need to inspect the predicates of each of the K CCs.

Consider the $BadPass_{2,3}$ CC. Since only player P_1 has the ball (see Fig. 1), $HasBall(P_2)$ is false and CC for $BadPass_{2,3}$ can be deactivated. We can also deactivate $BadPass_{x,y}$ CCs for other pairs of players where P_x does not have the ball, thus reducing the quadratic number of $BadPass_{x,y}$ CCs to a linear number of $BadPass_{1,y}$ CCs.

Another observation is that agents who are very far apart do not need to coordinate. We can define a $Nearby_{x,y}$ predicate that is true if two agents are within a given distance and conjunct it with those predicates involving them. This simple strategy has proven effective in reducing $\mathcal{A}_0$ for directing top level exploration in practice.

3. EXPERIMENT RESULTS

We carried out experiments to evaluate the proposed approach on two domains: simplified soccer (Fig. 1) and tactical real-time strategy (RTS) [4]. The environments are fully observable and episodic. All learning is online as no experience is saved and replayed. We compare four RL players. *Independent player* – each agent selects their own action independently and learns a separate policy [6]. *Flat player* – RL using single level on-policy learning with coordination graph (CG) defined by features for joint action selection [12]. *Solo player* – a representative of having individual task hierarchies for each agent. Only unary constraints are used. *Coordinated player* – uses our full two-level learning system. For function approximation, the solo and coordinated players use the same features as the flat player for their bottom level value function. Hence they have the same CGs defined by the features. The independent player has only features that involve single agents.

We design three types of experiments to investigate the performance of the two-level learning system. Each experiment progressively includes other methods that make RL in multi-agent domains practical. A video of the sample runs of our policies can be viewed at: `http://youtu.be/aloAOTBEUZ4`.

3.1 Simplified Soccer Domain

In soccer, the objective is to score the first goal in the shortest time. The soccer field is a *grid world*. Soccer players can stay, move in 4 directions, or the player with the ball may pass or shoot with a probabilistic chance of success weighted by distance. The ball changes ownership to the opposing team if the player with the ball collides with any player or if a pass fails. A failure to score a goal results in the ball going to the nearest player to the goal. In each time step, submitted player actions are randomly shuffled and executed. Rewards are 1 for winning, and -1 for losing.

There are three types of scripted opponents in order of difficulty: *random* players select actions at random; *defensive* players stay around the home quarter of the field and move to intercept the ball if it enters, the player with the ball does a solo counter-attack; *aggressive* players always go for the ball, once attained, the player with the ball heads for the goal while each of the other players stays near (marks) their respective enemy players. In soccer, good policies may require agents to have specific roles.

In addition to unary CCs, for pairwise CCs in soccer, we used a predicate for static collision CCs and three predicates for dynamic CCs including: $BadPass$, not jointly intercepting as opponent with the ball, and jointly blocking opponents' movements.

3.1.1 Only Exact Methods

For the first type of experiments, we examine our proposed two-level RL system without any approximation, using exact tabular functions and action selection via enumeration. This is to investigate if the extra top level is indeed useful for learning performance. A tabular function is equivalent to a linear function approximation where each feature is a boolean variable corresponding to an entry in the table. We only compare tabular flat and coordinated players.

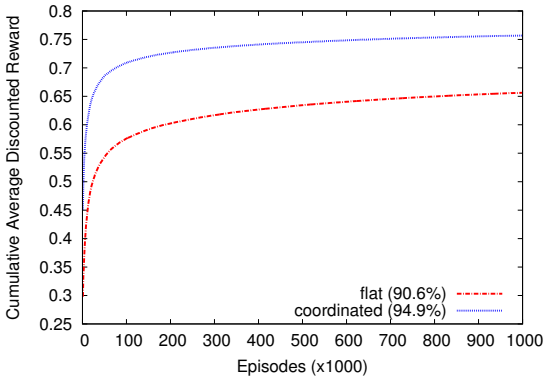
The soccer field is 6×4 units and the RL players have 2 soccer players versus 1 player for each type of scripted opponent. RL players used discounting ($\gamma = 0.99$), ϵ-greedy policies with constant $\epsilon = 0.1$, and constant step size ($\alpha = 10^{-3}$). The coordinated player used only pairwise CCs and no unary CCs for this experiment. Its top level has 3 dynamic CCs giving a top joint action space of size 2^3. With tabular functions the number of parameters to learn are more than 10,000 for each RL player. Hence we run the experiments for many episodes.

Fig. 4 shows the results for the three scripted opponents. 1 million episodes are used against random, and 10 million episodes against both defensive and aggressive opponents. From the results we see that learners have poorer goal achievement against harder opponents. The coordinated player performs consistently better than the flat player. This verifies that the top level of our system is stable and improves learning performance when there are no other factors.

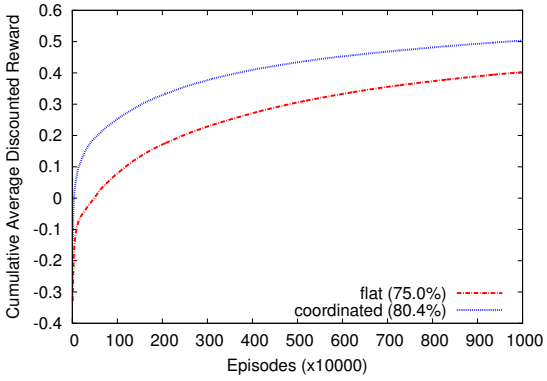
3.1.2 With Function Approximation

For the second type of experiments, we evaluate the RL system using propositional features (PFs) for linear value function approximation and exact action selection through bucket elimination. Top level action selection is independent (see Section 2.4). The soccer field is 12×8 units. The RL players have 4 soccer players versus 6 players each using the defensive and aggressive scripts. The size of the state space is at least 10^{13} and the size of the action space is 8^4. The top level has 6 static CCs and $18 (= 6 \times 3)$ dynamic pairwise CCs. The learners used discounting with $\gamma = 0.99$, and ϵ-greedy policies with decaying step size (α) and exploration (ϵ) parameters written as $param = \langle \text{initial, final, decay rate} \rangle$: all players used $\epsilon = \langle 1.0, 0.01, 0.998 \rangle$, the flat player used $\alpha = \langle 0.2, 0.1, 0.998 \rangle$, and the rest used $\alpha = \langle 0.1, 0.01, 0.998 \rangle$.

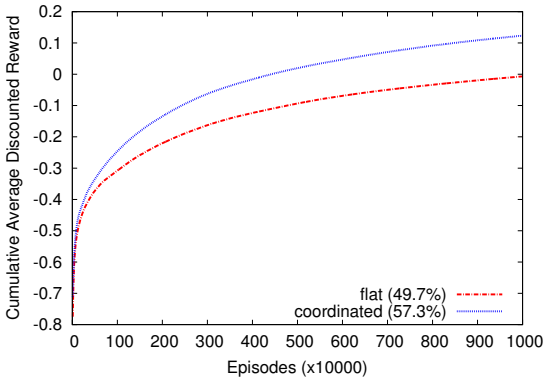
Fig. 5 shows the cumulative average discounted rewards for the soccer setup against the two different scripted strategies. Note that the plots start from the tenth episode. For all opponents, the flat player performs better than the independent player, indicating that coordination is important for simplified soccer. The solo and coordinated players' policies converged and performs better than the flat and independent players. This shows that our proposed system results in better policies than flat RL with coordination graphs. The coordinated player performs better than the solo player against both the defensive and aggressive opponents. Its benefit came from good early exploration compared to the other RL players. This indicates that CCs are effective in



(a) Random, 2 v. 1 soccer players, 1M episodes



(b) Defensive, 2 v. 1 soccer players, 10M episodes



(c) Aggressive, 2 v. 1 soccer players, 10M episodes

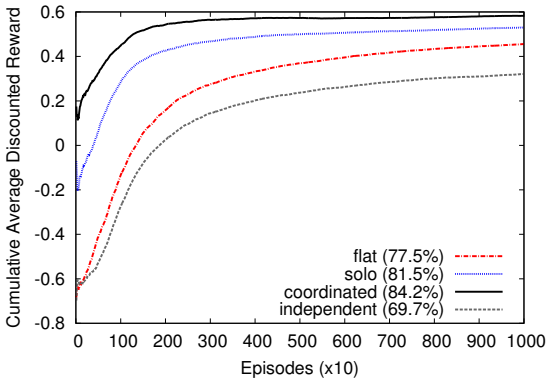
(Final win rates in brackets)

Figure 4: Soccer results for Section 3.1.1, each plot averaged over 10 runs.

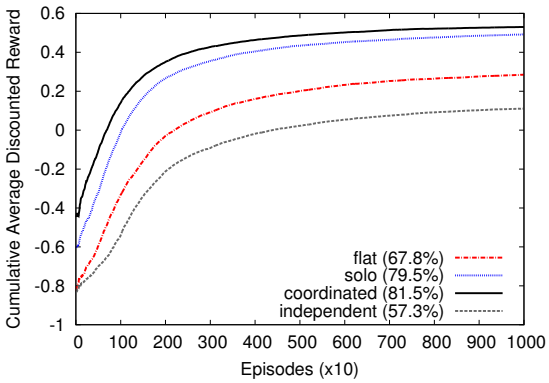
the online setting where it is generally harder to improve better policies due to the exploration-exploitation trade-off.

3.2 Tactical RTS Domain

The goal in tactical RTS is to eliminate the enemy team of marines quickly in a 240×240 *point based* map. Each marine occupies a point on the map with a fixed radius and a number of hit points. When its hit points reaches zero, it is destroyed. A marine's action domain consists of the 8 compass directions, an attack action for each possible enemy, and idle. The size of the action space is at least 10^{10} while the huge state space consists of all the possible marines' positions and hit points.



(a) Defensive, 4 v. 6 soccer players.



(b) Aggressive, 4 v. 6 soccer players.

(Final win rates in brackets)

Figure 5: Soccer results for Section 3.1.2. 10K episodes averaged over 10 runs each.

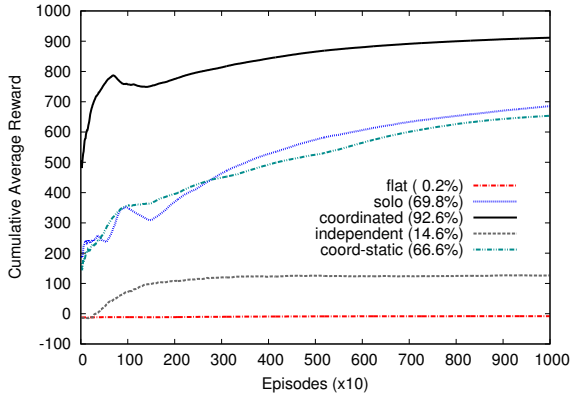
We pit the RL players against two scripted opponents: *aggressive* marines head for the nearest enemy and shoot enemies in range, *unpredictable* marines move in random directions and shoot enemies in range. The unpredictable opponent may be strong if its marines move in the same direction towards the enemy, or weak if they scatter. Opponents' marines are able to shoot and move at the same time giving them an advantage. RL players must quickly learn to shoot and exploit teamwork as marines die easily. This makes it difficult to explore winning episodes. Rewards are -0.1 per time step and 10^3 for opponent team elimination.

3.2.1 *Relational Features & Inexactness*

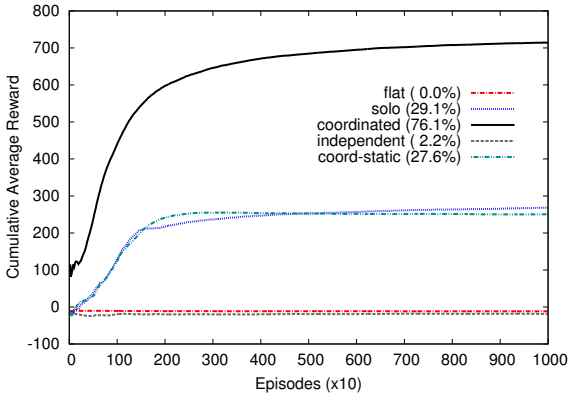
The final type of experiment integrates methods to deal with large multi-agent problems where the number of agents change over time. We incorporate the use of relational features (RFs) for generalization of learning and approximate primitive action selection using the Max-plus algorithm.

Four setups are used in our experiments: (a) 10 RL marines versus 10 aggressive marines, (b) 10 RL marines versus 13 unpredictable marines, (c) 10 RL marines versus 13 aggressive marines, (d) 10 RL marines versus 5 unpredictable super marines. The super marines in setup (d) have twice the firepower and hit points, hence coordination for the RL players is very crucial for success.

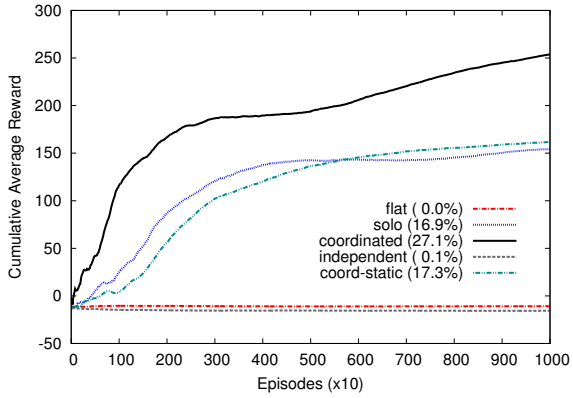
For each setup, the RL players use $\gamma = 1$ with the same decaying parameters. Setup (a), (c), (d) used $\epsilon = \langle 1, 0.01, 0.998 \rangle$, $\alpha = \langle 0.01, 10^{-4}, 0.998 \rangle$, while (b) used $\epsilon = \langle 1, 0.1, 0.998 \rangle$, $\alpha = \langle 0.01, 10^{-6}, 0.998 \rangle$. Setup (b) was given more



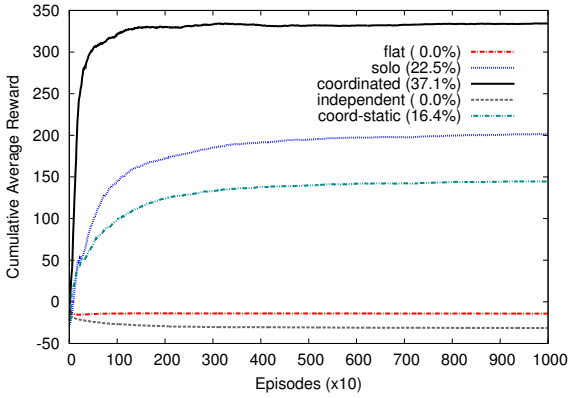
(a) Aggressive, 10 v. 10 marines.



(b) Unpredictable, 10 v. 13 marines.



(c) Aggressive, 10 v. 13 marines.



(d) Unpredictable, 10 normal v. 5 super marines.

(Final win rates in brackets. Approximate action selection using Max-plus.)

Figure 6: RTS results for Section 3.2.1. 10K episodes averaged over 10 runs each.

exploration due to the unpredictable nature of the opponent. Agents need to coordinate if they are within 30 points of each other, otherwise their pairwise features are set to zero. RFs are useful for RTS as the number of agents vary over time, except for the independent player that learns separate policies. We used 45 static collision CCs and 90 dynamic CCs from predicates for troop formation to maximize overlapping firepower, and to protect wounded teammates.

Fig. 6 shows the cumulative average reward for the RTS setups. We also included a new RL player, *coord-static*, which utilized only the static collision CCs in addition to the capabilities of the solo player. Total percentage wins are shown in brackets. We observe that all the RL players' policies converged over time. As seen in soccer, all two-level players: solo, coord-static, and coordinated; out-performed existing approaches of independent, and flat players. The coordinated player was able to learn quickly in all the four setups. The flat player experienced few winning episodes and ended up trying to lose as fast as possible to reduce the total negative reward obtained from each time step. The independent player managed to learn some strategy in Fig. 6a and wins more episodes than the flat player in the other setups. However, its reward is less than the flat player in those setups.

The results for the coord-static and solo players are mostly comparable. This is because the coord-static's marines tend to spread out more often and are destroyed easily when isolated. This occurs while ϵ is high and it has yet to learn a good formation. After sufficient exploration, the coord-static player is competitive with the solo player, although it learned a poorer policy in (d).

Conversely, the performance gains by the coordinated player with dynamic pairwise CCs are large compared to the others. Hence CCs are obviously crucial for tactical RTS. It is clear that most learning benefits came from the dynamic CCs. The coordinated player has more coordination than the solo player. This is also confirmed in our video which shows the coordinated player overcoming the enemy force simply by having better coordination among its marines. The results indicate that allowing the RL system to guide its exploration via dynamic CCs is effective for improving the learning rate in MDPs with large joint action spaces.

4. DISCUSSION & RELATED WORK

Previous works in task-based RL for multiple agents [15, 9, 16] require users to define tasks, terminating conditions, reward decomposition among tasks and agents, and new features for every level in the task hierarchy to represent them. They learn high level actions to constrain the exploration of the original MDP's actions based on single agent tasks. But it is not straightforward to incorporate coordination among agents to aid exploration. In contrast, the proposed two-level RL system employs declarative CCs and allows existing predicates for features to be reused as CCs to guide itself. Our results show that the two-level learning system outperforms task-based RL with coordination graphs when

comparing their ability at constraining exploration. This demonstrates that dynamic multi-agent CCs are important.

The work in [16] presented a two-level method where the top level assigns tasks and the bottom level learns with the task restrictions. Our work differs as our top level explores CC activations that are defined on multiple agents, and learns a value function that eliminates the need for a costly nested maximization when selecting CCs to activate. The proposed CCs are distinct from methods that learn coordination structure [13] within the value functions themselves. In our work, we use CCs to direct exploration by specifying subsets of the joint action space to be pruned. This is dynamically learned by our two-level system. The work in [5] presented a method that used constraints involving multiple agents. They require an offline phase for learning with constraints, and the constraints are static. Our work is fully online from the onset and learns to use CCs in different states. In [21], fixed heuristic supervisor agents biased base agents' policies with a coarse-grained approach. In contrast, ours employs fine-grained RL at the top level using the original reward signal and state observations.

Another branch of works deal with zero communication multi-agent problems known as Markov games where the focus is on handling the non-stationary environment due to independent learning and the setting is mostly adversarial [22]. In [3], heuristics can be provided to influence learning when the policy selects a maximal action. Their heuristics do not affect exploratory actions and are used in an adversarial setting with a much smaller action space.

5. CONCLUSION

We have investigated the use of expert coordination knowledge to improve RL via CCs for multi-agent MDPs from a centralized perspective. The proposed system's top level learns to activate CCs to guide the bottom level's exploration towards better experience. Learning to activate CCs allows flexibility in discovering good policies. Conversely, having only static CCs may lead to over-constraining the policy. We conducted experiments that progressively integrate our system with other useful methods for multi-agent problems. Our results on different domains demonstrate that the two-level RL system leads to better policies compared to existing approaches. Further, RL with CCs makes better use of early exploration, especially with multi-agent CCs. This is advantageous for online applications as overall higher goal achievement is attained. Future work involves automating the construction of CCs to reduce reliance on expert knowledge. Others include decentralized learning for distributed applications with communication costs, and fusing our method with task-based methods.

6. ACKNOWLEDGMENTS

This work was supported by A*STAR Exploit Flagship Grant ETPL/10-FS0001-NUS0.

7. REFERENCES

[1] D. Andre and S. J. Russell. State abstraction for programmable reinforcement learning agents. In *AAAI*, pages 119–125, 2002.

[2] N. Asgharbeygi, D. J. Stracuzzi, and P. Langley. Relational temporal difference learning. In *ICML*, pages 49–56, 2006.

[3] R. A. C. Bianchi, C. H. C. Ribeiro, and A. H. R. Costa. Heuristic selection of actions in multiagent reinforcement learning. In *IJCAI*, pages 690–696, 2007.

[4] M. Buro. Call for AI research in RTS games. In *AAAI Workshop on AI in Games*, pages 139–141. AAAI Press, 2004.

[5] G. Chen, Z. Yang, H. He, and K. M. Goh. Coordinating multiple agents via reinforcement learning. *AAMAS*, 10:273–328, 2005.

[6] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI*, pages 746–752, 1998.

[7] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *JAIR*, 13:227–303, 2000.

[8] P. Geibel. Reinforcement learning for mdps with constraints. In *ECML*, volume 4212 of *LNCS*, pages 646–653, 2006.

[9] M. Ghavamzadeh, S. Mahadevan, and R. Makar. Hierarchical multi-agent reinforcement learning. *AAMAS*, 13(2):197–229, 2006.

[10] C. Guestrin, D. Koller, C. Gearhart, and N. Kanodia. Generalizing plans to new environments in relational mdps. In *IJCAI*, pages 1003–1010, 2003.

[11] C. Guestrin, D. Koller, and R. Parr. Multiagent planning with factored mdps. In *NIPS*, pages 1523–1530, 2001.

[12] C. Guestrin, M. G. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *ICML*, pages 227–234, 2002.

[13] J. R. Kok, P. Jan, H. Bram, and B. N. Vlassis. Utile coordination: Learning interdependencies among cooperative agents. In *IEEE Sym. on CIG*, pages 29–36, 2005.

[14] J. R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.

[15] B. Marthi, D. Latham, S. Russell, and C. Guestrin. Concurrent hierarchical reinforcement learning. In *IJCAI*, pages 779–785, 2005.

[16] S. Proper and P. Tadepalli. Solving multiagent assignment markov decision processes. In *AAMAS*, volume 1, pages 681–688, 2009.

[17] R. Stranders, F. M. D. Fave, A. Rogers, and N. R. Jennings. A decentralised coordination algorithm for mobile sensors. In *AAAI*, pages 874–880, 2010.

[18] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[19] R. S. Sutton, D. Precup, and S. P. Singh. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[20] P. Tadepalli, R. Givan, and K. Driessens. Relational reinforcement learning: An overview. In *ICML'04 Workshop on Relational Reinforcement Learning*, pages 1–9, 2004.

[21] C. Zhang, S. Abdallah, and V. Lesser. Integrating organizational control into multi-agent learning. In *AAMAS*, volume 2, pages 757–764, 2009.

[22] C. Zhang and V. R. Lesser. Multi-agent learning with policy prediction. In *AAAI*, pages 927–934, 2010.

On Coalition Formation with Sparse Synergies

Thomas Voice, Sarvapali D. Ramchurn, Nicholas R. Jennings
Agents, Interaction and Complexity Group
School of Electronics and Computer Science
University of Southampton, UK
{tdv,sdr,nrj}@ecs.soton.ac.uk

ABSTRACT

We consider coalition formation problems for agents with an underlying *synergistic graph*, where edges between agents represent some vital synergistic link, such as communication, trust, or physical constraints. A coalition is infeasible if its members do not form a connected subgraph, meaning parts of the coalition are isolated from others. Current state-of-the-art coalition formation algorithms are not designed for problems over synergistic graphs. They assume that *all* coalitions are feasible and so involve redundant computation when this is not the case. Accordingly, we propose algorithms, namely D-SlyCE and DyCE, to enumerate all feasible coalitions in a distributed fashion and find the optimal feasible coalition structure respectively. When evaluated on a variety of synergistic graphs, D-SlyCE is up to 660 times faster while DyCE is up to 7×10^4 times faster than the state-of-the-art algorithms. For particular classes of graphs, D-SlyCE is the first to enumerate valid coalition values for up to 50 agents and DyCE is the first algorithm to find the optimal coalition structure for up to 30 agents in minutes as opposed to months for previous algorithms.

Categories and Subject Descriptors

I.2.11 [**Distributed AI**]: Multi-Agent Systems

General Terms

Algorithms

Keywords

Coalition Formation, Networks

1. INTRODUCTION

Coalition formation is one of the fundamental approaches in multi-agent systems for establishing collaborations among agents, each with individual objectives and properties [10]. Key computational tasks in coalition formation involve calculating the value of each potential coalition and finding the best set of coalitions to be formed (i.e., the coalition structure generation problem) [9]. To date, existing work has typically studied coalition formation in abstract settings, where every set of agents can be considered to be a potential coalition [3, 5, 7, 10]. Such approaches are limited to solving problems involving at most 30 agents (28 in the case of coalition structure generation). However, this may be severely inefficient for problems where only some coalitions are feasible. We believe that many real-world applications involve constraints as to which coalitions can exist based on sparse synergies (i.e., necessary peer-to-peer connections) between individual agents in the system [2, 8]. These constraints may be due to communication constraints (e.g., non-overlapping communication loci or energy limitations for sending messages across a network), social or trust relationships (e.g., energy consumers who prefer to group with their friends and relatives in forming energy cooperatives), or physical constraints (e.g., emergency responders that have enough fuel to join only specific teams or have life-saving capabilities that match only a limited number of other responders).

Against this background, in this paper we provide efficient algorithms to form coalitions in environments that have an underlying *synergistic graph*, where edges in the graph represent vital synergistic links between agents that constrain which coalitions may form. In more detail, we consider coalition formation problems where each coalition is only feasible if its members form the vertices of a connected subgraph of the constraining graph. By taking advantage of these structures, we aim to improve on the performance of the current state-of-the-art coalition formation algorithms which were not designed with synergistic graphs in mind.

In this setting,[1] we advance the state-of-the-art in the following ways. First, we provide a new exact coalition enumeration procedure, SlyCE (Sequentially connected Coalition Enumeration). Second, we provide D-SlyCE (Distributed SlyCE), a variant which aims to evenly distribute the SlyCE computation amongst agents at negligible communication and computation cost. Third, building upon SlyCE, we provide a complete algorithm, DyCE (Dynamic programming for optimal connected Coalition structure Evaluation) to find the optimal coalition structure. Fourth, we benchmark our solutions against the state-of-the-art algorithms and show that D-SlyCE can be up to 660 times faster and DyCE can be up to 7×10^4 times faster. Moreover, for particular classes of graphs, our algorithms are the first to be able to enumerate coalition values for 50 agents and find the optimal coalition structure for up to 30 agents within minutes compared to months for the state-of-the-art.

The rest of the paper is structured as follows. In Section 2 we discuss related work. Then, in Section 3, we formally describe the problem of coalition formation with sparse synergies. In Section 4 we describe SlyCE and discuss its prop-

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

[1]The reader is referred to [2, 4] for a cooperative game theoretic analysis (which is not the goal of this paper) of our setting for the case where agents are self-interested.

erties. We then propose the D-SlyCE algorithm in Section 5, to distribute the computation of SlyCE amongst agents fairly. Then, we turn to the coalition structure generation problem in Section 6 and describe our solution, DyCE. We empirically benchmark D-SlyCE and DyCE in Section 7. Lastly, Section 8 concludes.

2. BACKGROUND

The formation of coalitions within synergistic graphs (as defined in this paper) has typically been studied in the field of economics where the focus is on the definition of cooperative game-theoretic solutions [2, 4]. In contrast, this paper is concerned specifically with the coalition value calculation and coalition structure generation problems over synergistic graphs.

The challenge in coalition value calculation is to enumerate all the feasible coalitions and efficiently distribute this computation among the agents. The main algorithms that deal with this specifically are Shehory et al.'s [10] and DCVC [5]. The latter is the fastest and is able to enumerate coalitions for up to 30 agents in reasonable time. Under DCVC, for each $s = 1, \dots n$, each agent calculates the coalition values for an nth share of a lexicographically ordered list of all coalitions of size s. DCVC does this in such a way that every coalition value is calculated precisely once, and no agent communication is required. While DCVC also efficiently recomputes coalition values where individual agents may be removed or new agents added dynamically, it has no way of avoiding infeasible coalitions.

Turning to the coalition structure generation problem (effectively a set partitioning problem), a number of recent works in this area have attempted to solve the problem in both centralised and distributed ways along with providing anytime quality guarantees [9, 6, 3, 7]. In particular, we note the two approaches taken in this area (i) low-complexity ($O(3^n)$) complete algorithms based on dynamic programming, such as DP and IDP, that have guaranteed run-times for arbitrary coalition value distributions (ii) high worst-case complexity ($O(n^n)$) complete algorithms, based on branch-and-bound techniques, that have anytime properties but heavily depend on the coalition value distribution in order to establish bounds on segments of the search space and therefore prune huge parts during the search process. While the latter algorithms have been shown to be faster than the former given specific coalition value distributions, their approach is undefined for cases where only some coalitions are feasible. Simply treating infeasible coalitions as being feasible but with value $-\infty$ would not be suitable, as they use averages of coalition values to compute lower bounds. In contrast, IDP (the fastest dynamic programming approach) makes no requirements on coalition value distributions and thus, we can treat infeasible coalitions as being feasible coalitions with value $-\infty$. This has no effect on the runtime of IDP, which only depends on n. We therefore use IDP to benchmark DyCE.

Another candidate set of techniques to solve the coalition structure generation problem is Integer Programming based solvers. These have been shown to be particular inefficient when all coalitions are deemed feasible (due to the size of the input) but tend to be very efficient in solving other combinatorial optimisation problems (combinatorial auctions or set packing problems). Hence, we also benchmark DyCE against IBM's ILOG CPLEX in Section 7.

Finally, another related algorithm to ours is Rahwan et al.'s CCF algorithm [8] which does solve coalition structure generation problems with constraints on which coalitions can be formed but they consider a different constraint model to ours.

3. MODEL

In this section, we model the problem of Coalition Formation with Sparse Synergies (CFSS). We identify a set of agents with the set of positive integers $I = \{1, 2, ..., n\}$, where n is the total number of agents in the system. Agents can form coalitions $C \subseteq I$, however the set of feasible coalitions, $\mathcal{C}$, is constrained by a graph $G = (I, E)$, where E is a set of edges between agents. We consider the situation where a coalition of agents C is feasible if and only if there exists a connected subgraph $G' = (C, E')$ with edges $E' \subseteq E$ and vertex set exactly equal to C. By forming coalitions, agents can perform tasks in the environment and their effectiveness in performing such tasks depends on the synergies in their abilities generated by them being in the same coalition. To capture such synergies, we define the value of a coalition using a characteristic function $v : \mathcal{C} \rightarrow \mathbb{R}$. The function $v(\cdot)$ may be arbitrarily defined according to the domain, however coalition values are always independent of any other coalitions that may exist. We address the problem of enumerating and evaluating all feasible coalitions for CFSS problems by providing the SlyCE and D-SlyCE algorithms, in Section 4. Now, given the coalition values, a key challenge is to choose the best *coalition structure*, that is, the best set of disjoint coalitions that collectively cover all agents. This problem is termed the coalition structure generation problem. To find the optimal coalition structure in CFSS, we need to consider the set of *feasible coalition structures*, $\mathcal{F}(G)$, that is, the set of coalition structures that only contain feasible coalitions. The coalition structure generation problem which DyCE, given in Section 6, attempts to solve is then to find $\arg\max_{CS \in \mathcal{F}(G)} \sum_{C \in CS} v(C)$. In the next section, we proceed with our description of the SlyCE and D-SlyCE algorithms.

4. THE SLYCE ALGORITHM

The SlyCE algorithm proceeds by conducting a depth first search over a tree representation of the set of feasible coalitions in G. At each point in the search, SlyCE maintains a root node, R, which represents the current coalition being evaluated, and a frontier set, F, that contains agents that may be added to the root node to form a feasible coalition. The individual steps of SlyCE are described in Algorithm 1, where we use the function $\overline{N}(\cdot, \cdot)$ which we define to be:

$$\overline{N}(F, R) = \{j | j > \min(R \cup F)\} \cap N(F) \setminus (N(R) \cup R \cup F),$$

for $N(\cdot)$ denoting the set of neighbours of a subset of I in G, that is, $N(R) = \{j : \exists i \in R, (i, j) \in E\}$. The algorithm recursively traverses the search tree in two phases. The first phase, (lines 2 and 3), involves generating a new root node R by combining the current root node with a subset of the frontier set. Having updated the root node, (and evaluated it on line 4), in the second phase, (line 5), a new frontier set is created by choosing neighbouring agents to expand the root with. The algorithm then calls itself (line 6) to continue the recursive search. The key point here is that SlyCE chooses those agents in the root creation and expansion phases so

that it will not recompute an already computed coalition. We elaborate on these two phases next.

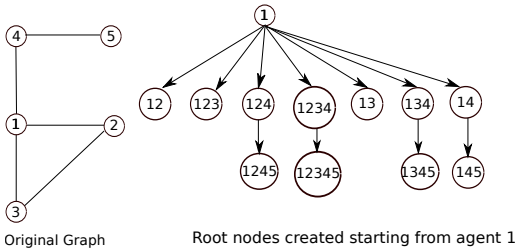


Figure 1: Root expansion process emanating from agent 1.

1. Root expansion — given an existing root $R \subseteq I$ and a frontier set $F \subseteq I$, form a new root node R' such that $R' = R \cup F^*$ where $F^* \subseteq F$. Simply put, given an existing root node, we expand it to include agents that are not currently included and that lie at the frontier of the existing root. Figure 1 shows an example of this.

2. Frontier expansion — given a new root node $R' = R \cup F^*$ formed from existing root node $R \subseteq I$ and set F^*, we set the new frontier set F' to be the set of agents j such that, $j > \min(R')$, $j \notin R'$, $j \notin N(i)$ for all $i \in R$, and $j \in N(k)$ for some $k \in F^*$. That is, the ids of all agents in F' should be numerically higher than the id of the agent with the lowest id in R', and F' should consist of agents who are neighbours of F^*, but are neither neighbours nor members of R'.

Figure 1 shows how SlyCE expands to other root nodes, starting with $R = \{1\}$. These are evaluated and their value stored. The same process, starting from the other agents, will generate totally different coalitions (e.g., 5 will only generate $\{5\}$, while 4 will generate $\{4\}$ and $\{4, 5\}$).

Algorithm 1 $slyce(R, F, m)$

```
1: if F ≠ ∅ and m > 0 then
2:    for all F* ⊆ F with 1 ≤ |F*| ≤ m do
3:       R' ← F* ∪ R {Generate new root node.}
4:       compute and store v(R') {Evaluate new root node.}
5:       F' ← N̄(F*, R) {Generate new frontier set}
6:       slyce(R', F', m − |F*|) {Recursive call.}
7:    end for
8: end if
```

In order to enumerate all coalitions of size up to m, SlyCE should search from each agent singleton, which can be achieved by calling $slyce(\emptyset, \{i\}, m)$ for all $i \in I$.

4.1 Properties of SlyCE

We now elaborate on the key properties of SlyCE, in particular its correctness, completeness, and non-redundancy. We first note that SlyCE only evaluates *feasible coalitions*. To be more precise, we claim that, provided that R is feasible and $F \subseteq N(R)$, then for any m, $slyce(R, F, m)$ only calls $v(C)$ for feasible coalitions $C \subseteq I$. To see this, note that for such F and R, for any $F^* \subseteq F$, $F^* \subseteq N(R)$ and so when $v(R')$ is evaluated, $R' = F^* \cup R$ is a feasible coalition. Furthermore, for $F' = \overline{N}(F^*, R)$, $F' \subseteq N(F^*) \subseteq N(F^* \cup R) = N(R')$, and so the recursive call to $slyce(R', F', m - |F^*|)$ also satisfies $F' \subseteq N(R')$. This recursively proves our claim. It remains to note that when $slyce(\emptyset, \{i\}, m)$ is called for some $i \in I$, after the first level of recursion, all subsequent F and R satisfy $F \subseteq N(R)$. Hence, SlyCE only evaluates feasible coalitions and is thus, *correct*. It is also important to show that SlyCE is *complete*, that is, that it can enumerate all feasible coalitions. We do so in the following proposition.

PROPOSITION 1. *For each $i \in I$, and $m \geq 1$, if the SlyCE algorithm is called with parameters $slyce(\emptyset, \{i\}, m)$ then every feasible coalition $C \subseteq I$ such that $i = \min(C)$ and $1 \leq |C| \leq m$ will be evaluated.*

PROOF. First, we should note that since at least one agent is added to R before it is passed to the next level of recursion, we have that at the lth level of recursion, $|R| \geq l$. Thus, the algorithm cannot go beyond the mth level of recursion, and must terminate. This means that if $slyce(R, F, k)$ is evaluated, the main **for all** loop reaches every subset of F of size up to k.

Now, suppose C is a feasible coalition with $i = \min(C)$ and $1 \leq |C| \leq m$. Let $E' \subseteq E$ be the set of all edges $(j, k) \in E$ such that $j, k \in C$. For all $j, k \in C$, let $d(j, k)$ be the number of vertices in the minimal path between j and k over the subgraph $G' = (E', C)$, and let s be the maximum value of $d(i, j)$ for $j \in C$. If $s = 1$ then $C = \{i\}$ and $v(C)$ is evaluated during the initial call of $slyce(\emptyset, \{i\}, m)$.

We now consider the case where $s > 1$. Let us define the sets $F_1, F_2, \ldots, F_s$ as being $F_l = \{j : d(i, j) = l\}$, for $l = 1, \ldots, s$, and $H_0, H_1, \ldots, H_s$ as being $H_0 = \emptyset$, $H_l = \{j : d(i, j) \leq l\}$ for $l = 1, \ldots, s$. Consider the set F_{l+1} for some $l = 1, \ldots, m - 1$. Since $i = \min(C)$, we must have that, for all $j \in F_{l+1}$, $j > i$. Furthermore, for all $j \in F_{l+1}$, j cannot lie in H_{l-1} nor can it be a neighbour of H_{l-1}, otherwise there would be a path shorter than $l+1$ from j to i through G'. So, the first step along the path of length $l + 1$ from j to i through G' must be from j to an agent in F_l and hence $j \in N(F_l)$. Thus, $F_{l+1} \subseteq \overline{N}(F_l, H_{l-1})$.

We now claim that during the operation of $slyce(\emptyset, \{i\}, m)$, $slyce(H_l, \overline{N}(F_l, H_{l-1}), m - |H_l|)$ will be called for all $l = 1, \ldots, s - 1$. This can be proved inductively. For $l = 1$, we have $H_1 = F_1 = \{i\}$, and $H_0 = \emptyset$. From the definition of the algorithm, $slyce(\{i\}, \overline{N}(\{i\}, \emptyset), m - 1)$ is called as part of the main loop of $slyce(\emptyset, \{i\}, m)$. Now suppose that for some $l < s$, $slyce(i, H_l, \overline{N}(F_l, H_{l-1}))$ is called. Since $F_{l+1} \cup H_l \subseteq C$, then $|F_{l+1}| + |H_l| \leq s$ and so $|F_{l+1}| \leq s - |H_l|$. As shown above, $F_{l+1} \subseteq \overline{N}(F_l, H_{l-1})$, and so $slyce(i, H_l \cup F_{l+1}, \overline{N}(F_{l+1}, H_l))$ must be called as part of the operation of $slyce(i, H_l, \overline{N}(F_l, H_{l-1}))$. As $H_l \cup F_{l+1} = H_{l+1}$ this proves our claim by induction. It remains to note, when $slyce(H_{m-1}, \overline{N}(F_{m-1}, H_{m-2}), m - |H_{m-1}|)$ is called, the value of $v(F_m \cup H_{m-1}) = v(C)$ is evaluated, as required. □

Thus, for every feasible coalition $C \subseteq I$, $v(C)$ is evaluated during the operation of $slyce(\emptyset, \{\min(C)\}, n)$. It is also true that SlyCE is *non-redundant*, that is, during its operation, it never evaluates the same coalition twice. We can see that this is true by noting that, when a subset $F^* \subseteq F$ is chosen during SlyCE and $R \cup F^*$ is set as a root, then, from that depth of recursion onwards, the root will always contain R, and thus the frontier nodes will never again contain any

agent in $F \subseteq N(R)$. If we have two paths through the root expansion process that diverge, at the point of divergence, they must choose different root expansions, meaning that at least one of the paths adds at least one agent that can never be added at a later stage in the other path. Thus, there is no way to reach the same coalition twice through the recursive calls in SlyCE.

So, if the SlyCE algorithm is run as $slyce(\emptyset, \{i\}, m)$ for each agent $i \in I$, then for every feasible coalition C ($|C| \leq m$), $v(C)$ is evaluated and stored precisely once, and no infeasible coalitions are evaluated. Indeed, this can be run in parallel with each agent $i \in I$ computing $slyce(\emptyset, \{i\}, m)$. Thus, SlyCE may be implemented as a distributed algorithm. Moreover we can use SlyCE to do more than simply enumerate all feasible coalitions. As we describe in the following subsection, we can use SlyCE to cycle through feasible coalitions that are subsets of a given feasible coalition, a process which will be useful in DyCE (see Section 6).

4.2 SlyCE Over Coalition Subsets

The SlyCE algorithm, as we have defined it, enumerates all feasible coalitions of size up to some limit m for a given graph $G = (I, E)$. However, given a feasible coalition $C \subseteq I$, we can also use SlyCE to enumerate all feasible coalitions that are subsets of C. We simply have to apply SlyCE to the subgraph that G induces over the nodes of C. Since this is, itself, a connected graph, the desirable properties of SlyCE discussed above will still hold. For notational convenience, in preparation for our later description of DyCE, we define the function $nextslyce(\cdot, \cdot, \cdot)$ to be such that, for feasible coalitions $C' \subseteq C \subseteq I$ and integer m, with $|C'| \leq m$, $nextslyce(C', C, m)$ returns the subset of C that would follow C' during the process of SlyCE iterating through all feasible coalition subsets of C of size at most m. Thus, if we begin with $C' = \{a\}$, for $a \in C$, then repeated calls to $C' \leftarrow nextslyce(C', C, m)$ will conduct the entire SlyCE iteration $slyce(\emptyset, \{a\}, m)$ over the subgraph induced by G on C. If C' is the last in this SlyCE iteration, we stipulate that $nextslyce(C', C, m)$ should return the empty set.

Having elaborated on the properties of SlyCE above, we next discuss how the SlyCE computation may be distributed more fairly amongst agents, using our D-SlyCE algorithm.

5. THE D-SLYCE ALGORITHM

As noted above, SlyCE may be run as a distributed algorithm by having each agent $i \in I$ compute $slyce(\emptyset, \{i\}, m)$. However, in that case, agents with higher ids will have fewer computations than those with lower ids. Indeed, the agent with the highest id, only computes the value of one coalition, the singleton containing itself (agent 5 in Figure 1). In contrast, the agent with the lowest id would evaluate all feasible coalitions that contain it. Accordingly, in this section, we propose the Distributed SlyCE algorithm (D-SlyCE), under which the SlyCE algorithm is distributed across the agents such that the task assigned to each agent is determined with no communication overhead. Furthermore it generates no repeated coalition evaluations and spreads tasks reasonably fairly amongst agents (as shown in Section 7).[2] The full operation of D-SlyCE is defined in Algorithm 2.

Algorithm 2 $dslyce(i, m)$

```
 1: compute and store v({i})
 2: x ← i − 1
 3: for all agents a ∈ I do
 4:    F ← N̄({a}, ∅) {Assign frontier set.}
 5:    for all r = 1 ..., min(|F|, m) do
 6:       j ← ⌊(|F| choose r) × x/n⌋ {Index of beginning of share.}
 7:       while j < ⌊(|F| choose r) × (x + 1)/n⌋ do
 8:          F' ← L(F, j + 1, r) {Set next addition to root.}
 9:          j ← j + 1
10:          compute and store v(F' ∪ {a}) {As in SlyCE.}
11:          slyce(F' ∪ {a}, N̄(F', {a}), m − (r + 1))
12:       end while
13:       x ← (x + 1) mod n {Rotate share allocation.}
14:    end for
15: end for
```

rithm 2. It uses some combinatorial functions, which we define here. We use $\binom{m}{r}$ to denote the number of subsets of size r that may be drawn from a set of m unique agents, so $\binom{m}{r} = m!/r!(m-r)!$. For any set of agents C, for any $r = 1, \ldots, |C|$ and $i = 1, \ldots, \binom{|C|}{r}$ we let $L(C, i, r)$ denote the ith element of the set of all subsets of $|C|$ of size r under the lexicographic ordering induced by the ids of the agents. For any such i, $L(C, i, r)$ can be found easily, and if $L(C, i, r)$ is known then $L(C, i + 1, r)$ can be found quickly, using known lexicographic techniques that are described in [5].

D-SlyCE is called with the agent's id and the maximum coalition size m it should generate (in a similar vein to DCVC), such that if all coalitions must be generated, $m = n$. It then divides up the computation of the SlyCE algorithm amongst the agents by splitting up the operation of the first two levels of recursion of $slyce(\emptyset, \{i\}, m)$ for all $i \in I$. This division is done first by having each agent $a \in I$ evaluate $v(\{a\})$ (line 1). Then, each agent $a \in I$ is apportioned an approximately equal share of the subsets of $\overline{N}(\{i\}, \emptyset)$ for each $i \in I$ (line 6–7), that a must evaluate as per the main loop in $slyce(\{i\}, \overline{N}(\{i\}, \emptyset), m-1)$ (lines 10–11). Since larger frontier sets are likely to represent a greater computational burden, in order to make sure a gets an approximately fair allocation of tasks, for each agent i, a is given a $1/n$ share of the lexicographically ordered list of subsets of $\overline{N}(\{i\}, \emptyset)$ of size r for all $r = 1, \ldots, |\overline{N}(\{i\}, \emptyset)|$ (lines 6–7). Each time this dividing process occurs, the algorithm deterministically rotates the order in which shares are allocated to agents (line 13). This is because lexicographically earlier frontier sets contain lower id numbered agents, and thus are likely to represent a greater computational burden. As the full operation of D-SlyCE covers precisely the calls to $v(\cdot)$ and $slyce(\cdot, \cdot, \cdot)$ as in the first two levels of recursion of $slyce(\emptyset, \{i\}, m)$, for all $i \in I$ (lines 10–11), then D-SlyCE covers exactly the same calls to $v(\cdot)$ as SlyCE. This means that D-SlyCE has the same desirable properties as SlyCE, that is, D-SlyCE is correct, complete and non-redundant.

An example of how the computational burden of SlyCE is

[2]D-SlyCE is suitable for problems where evaluating $v(\cdot)$ is not computationally intensive. If the operation of SlyCE is a negligible overhead compared to the evaluation of $v(\cdot)$ for every feasible coalition, then the fairest way to distribute these evaluations would be for each agent to use SlyCE to create lists of all feasible coalitions, grouped together by size, and then evaluate a fair share from each list.

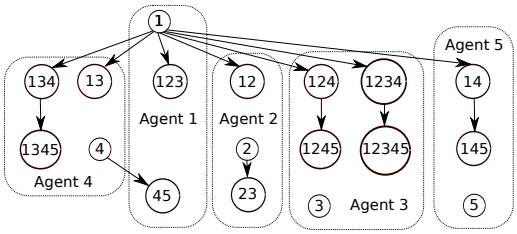


Figure 2: Distribution of coalition evaluation calculations under D-SlyCE.

divided up by D-SlyCE is given in Figure 2, which shows the distribution of coalition valuations prescribed by D-SlyCE for the graph given in Figure 1. Thus, we see that the first generation of descendants from each of the single node sets are divided up between the agents. Note that due to the way the shares of the lexicographical lists of subsets are rotated, agent 1 ends up computing $\{4, 5\}$. In general, the process results in a much fairer distribution than if each agent was simply allocated the frontier expansion tree that branches from their respective singleton coalition (i.e., $\{2\}$ and $\{2, 3\}$ for agent 2 or $\{5\}$ for agent 5 in SlyCE). Indeed, on complete graphs, D-SlyCE mimics the operation of DCVC, and thus is optimally fair.

Since single vertex root nodes with the lowest id represent the greatest computational burden, and the D-SlyCE distribution methods are most effective for root nodes with many frontier sets, we can further increase the fairness of the distribution by ensuring that the agents with highest degree have the lowest id number. This may be done in a fast, deterministic manner by each agent before it runs *dslyce*. In the next section, we move to solve the problem of coalition structure generation in synergistic graphs, using SlyCE as an important building block of DyCE.

6. THE DYCE ALGORITHM

In this section, we describe the operation of the DyCE algorithm, and prove its correctness. DyCE operates in a similar manner to IDP (see Section 2). However, it speeds up the search for the optimal coalition structure by using SlyCE to enumerate all feasible coalitions, thus allowing it to ignore infeasible coalitions during the search. DyCE further improves upon IDP by using SlyCE when evaluating feasible coalition subsets of coalitions (using the procedure described in Section 4.2).

DyCE solves a CFSS coalition structure generation problem by exploring the edges in the *feasible coalition structure graph* of the synergy graph. For any graph $G = (I, E)$, we define the *feasible coalition structure graph* of G, to be the directed graph with node set $\mathcal{F}(G)$, and directed edges $\mathcal{E}(G)$, where there is an edge from each $CS \in \mathcal{F}(G)$ to every $CS^* \in \mathcal{F}(G)$ that can be formed by splitting a coalition in CS into two smaller feasible coalitions. That is, $\mathcal{E}(G)$ is equal to the set of (CS, CS^*) where $CS \setminus \{C\} = CS^* \setminus \{C', C \setminus C'\}$ for some $C \in CS$ and some $C' \subset C$. Crucially, to improve tractability, we consider a reduced set of edges, $\mathcal{E}^*(G) \subset \mathcal{E}(G)$, where for $(CS, CS^*) \in \mathcal{E}(G)$ with $CS \setminus \{C\} = CS^* \setminus \{C', C \setminus C'\}$, $(CS, CS^*) \in \mathcal{E}^*(G)$ if $|C'| \leq n - |C|$ or $C = I$. In [7], a similar coalition structure graph is considered, and it is shown that it is possible to reach any coalition structure from $\{I\}$ along a similar reduced set of edges. DyCE requires the corresponding result to be true for our feasible coalition structure graph. This cannot be derived from the result in [7] because, although their set of reduced edges is similar to ours, paths along their coalition structure graph may pass through intermediary coalition structures that contain *infeasible coalitions* and thus are not in $\mathcal{F}(G)$. Instead, we prove the desired result in the following theorem.

THEOREM 1. *For every feasible coalition structure $CS \in \mathcal{F}(G)$, there is a directed path of edges from $\mathcal{E}^*(G)$ that leads from $\{I\}$ to CS.*

PROOF. Let us suppose this result does not hold. Let CS be the coalition structure with minimal $|CS|$ out of those $CS \in \mathcal{F}(G)$ that cannot be reached from $\{I\}$ by following the directed edges in $\mathcal{E}^*(G)$. We must have that $|CS| > 1$, otherwise CS would equal $\{I\}$. Further, $|CS| > 2$, as otherwise $(\{I\}, CS)$ would be in $\mathcal{E}^*(G)$.

Let C be the coalition in CS with maximal $|C|$. Suppose there were two coalitions $C', C'' \in CS \setminus \{C\}$ such that there is at least one edge in G between C' and C''. Let CS^* be,

$$CS^* = \{C' \cup C''\} \cup CS \setminus \{C', C''\}.$$

Since there is at least one edge between C and C', $C' \cup C''$ is feasible and so $CS^* \in \mathcal{F}(G)$. Since $|CS^*| < |CS|$ we must be able to get from $\{I\}$ to CS^* by following edges in $\mathcal{E}^*(G)$, by choice of CS. However, since $|C'| \leq |C|$ and $C \subset (I \setminus (C' \cup C''))$ we must have $|C'| \leq n - |C' \cup C''|$ and so $(CS^*, CS) \in \mathcal{E}^*(G)$. This leads to a contradiction, as we can now go from $\{I\}$ to CS^* and then to CS.

Thus, no two coalitions in $CS \setminus \{C\}$ have any edges between them. Since G is connected this means that there must be at least one edge between C and each coalition in $CS \setminus \{C\}$. Now, let C' be the coalition in $CS \setminus \{C\}$ with minimal $|C'|$. Let CS^* be,

$$CS^* = \{C \cup C'\} \cup CS \setminus \{C, C'\}.$$

Since there is at least one edge between C and C', $C \cup C'$ is feasible and so $CS^* \in \mathcal{F}(G)$. As $|CS^*| < |CS|$, by choice of CS there must be a directed path from $\{I\}$ to CS^* consisting of edges from $\mathcal{E}^*(G)$. As noted above, $|CS| > 2$, and so there must be some $C'' \in CS \setminus \{C, C'\}$. Since C'' is disjoint from $C \cup C'$, we must have $n - |C \cup C'| \geq |C''| \geq |C'|$, by choice of C'. Thus, $(CS^*, CS) \in \mathcal{E}^*(G)$, which leads to a contradiction, as now we can go from $\{I\}$ to CS^* and then to CS. Hence, no such CS can exist and the result follows. □

In order to explain how the feasible coalition structure graph relates to the operation of DyCE, we first need some definitions. For a graph $G = (I, E)$, and for any coalition structure $CS \in \mathcal{F}(G)$, let $\mathcal{D}(CS)$ be the set of coalition structures that can be reached from CS along directed paths made up of edges from $\mathcal{E}^*(G)$ (including CS itself). Theorem 1 shows that $\mathcal{D}(\{I\}) = \mathcal{F}(G)$. For a graph $G = (I, E)$ and a coalition valuation function $v(\cdot)$, we define the function $w : \mathcal{C} \to \mathbb{R}$ so that for any feasible coalition C, $w(C)$ is set equal to the maximum of: $\sum_{C' \in CS^* : C' \subseteq C} v(C')$, over $CS^* \in \mathcal{D}(CS)$, for $CS \in \mathcal{F}(G)$ such that $C \in CS$. Informally, if C is in some feasible coalition structure CS, then $w(C)$ is the maximum total value of subsets of C for coalition structures in $\mathcal{D}(CS)$. This is well defined and does not depend on choice of CS containing C because the set of splits that can occur as you move along edges in $\mathcal{E}^*(G)$

only depend on the coalition being split. From Theorem 1, $w(I)$ must be equal to the maximum of $\sum_{C' \in CS} v(C)$ over all $CS \in \mathcal{F}(G)$. So, $w(I)$ is the optimal feasible coalition structure value. For notational convenience if $C \subset I$ is not a feasible coalition then we let $w(C) = -\infty$.

The full operation of DyCE is described in Algorithm 3. DyCE proceeds by recursively calculating $w(\cdot)$ for all feasible coalitions, in increasing order of size. First, a memory block large enough to contain all coalitions of I is initialised, with every entry being given the value $-\infty$ (Line 1–3). Then, DyCE uses SlyCE to evaluate each feasible coalition (using procedure *nextslyce* described in Section 4.2) and its value is stored in memory, (lines 5–9). After initialisation, DyCE goes through each coalition of size s for $s = 1, 2, \ldots n$ (lines 10–29). For each coalition C of size s, if C is feasible, then DyCE calculates $w(C)$ and replaces the value of $v(C)$ in memory with $w(C)$. In order to calculate $w(C)$ for a coalition $C \neq I$, DyCE uses SlyCE to cycle through every feasible coalition subset $C' \subset C$ that is smaller than $n-|C|$ and $|C|/2$ (lines 11–14), and evaluates $w(C')+w(C \setminus C')$ (line 24). The value of $w(C)$ is then the maximum of these values and $v(C)$ (line 23). Note, we only go up to subsets C' of size $|C|/2$ as, for larger C', if $C \setminus C'$ is feasible then $w(C \setminus C')+w(C')$ will be evaluated anyway. DyCE also records the most recent set $C' \subset C$ such that $w(C) = w(C') + w(C \setminus C')$, if such exists. Lastly, $w(I)$ is calculated similarly by cycling through subsets $C \subset I$ with $|C| \leq n/2$ and maximising $w(C)+w(I \setminus C)$.

The optimal coalition structure is then determined (using Algorithm 4) from $w(I)$ by starting with $CS = \{I\}$ and then recursively replacing each coalition $C \in CS$ with C' and $C \setminus C'$ where $w(C) = w(C') + w(C \setminus C')$, if such exist. Note, if no such replacement is possible for a coalition $C \in CS$, then we must have that $w(C) = v(C)$. Furthermore, each time a replacement occurs, $\sum_{C \in CS} w(C)$ does not change. Hence the resulting coalition structure will be such that $w(C) = v(C)$ for all $C \in CS$ (since the algorithm only stops when no further subdivisions are possible) and $\sum_{C \in CS} v(CS) = \sum_{C \in CS} w(CS) = w(I)$. Hence CS is optimal. We can reduce memory requirements by not storing the particular subsets required to calculate this optimum, but instead, finding them once the values of $w(\cdot)$ have been found by searching through the subsets, as in [7].

To give an example of DyCE in operation, consider the graph given in Figure 1 with some coalition valuation function $v(\cdot)$. DyCE begins by using SlyCE to go through each feasible coalition C, evaluating $v(C)$ and recording the result. Then it goes through all sets of size $m = 1, 2, 3, 4, 5$ evaluating $w(C)$ for each feasible C, ignoring C if it is infeasible (i.e., if $v(C)$ was not recorded during the initial phase). For $|C| = 1$, $w(C) = v(C)$, and so this phase does not require any work. For $1 < |C| \leq 4$, DyCE must go through some subsets of C in order to evaluate $w(C)$. DyCE has to evaluate for subsets of size up to $\min(|C|/2, n - |C|)$ which is < 2 for such C. Thus $w(C)$ is the maximum of $v(C)$ and $w(\{a\}) + w(C \setminus \{a\})$ for each agent $a \in C$ such that $C \setminus \{a\}$ is feasible. Hence, for example, $w(\{1, 4, 5\}) = \max(v(\{1, 4, 5\}), w(\{1\})+w(\{4, 5\}), w(\{5\})+w(\{1, 4\}))$. For $C = \{1, 2, 3, 4, 5\} = I$, we evaluate $w(C') + w(C \setminus C')$ for all feasible coalitions C' such that $I \setminus C'$ is also a feasible coalition and $|C'| \leq n/2 = 2.5$. This is the same as evaluating $w(C') + w(C'')$ for all disjoint pairs of feasible coalitions C', C'' with $C' \cup C'' = I$. To give an example of why this process works, let us consider the set $C = \{1, 2, 3, 4\}$. Now,

Algorithm 3 *dyce*()

```
 1: for all C ⊂ I do
 2:   W(C) ← −∞ {Initialise memory.}
 3: end for
 4: for all a ∈ I do
 5:   C ← {a}
 6:   while C ≠ ∅ do
 7:     W(C) ← v(C) {Store coalition value.}
 8:     B(C) ← ∅ {Initialise best subset.}
 9:     C ← nextslyce(C, I, n) {Get the next coalition generated by SlyCE.}
10:   end while
11: end for
12: for all s = 1 ... n do
13:   m ← ⌊s/2⌋
14:   if s < n then
15:     m ← min(m, n − s)
16:   end if{Maximum subset size}
17:   for i = 1, ... (n s) do
18:     C ← L(I, i, s) {Go through sets of size s.}
19:     if W(C) > −∞ then {Ignore C if infeasible}
20:       for all a ∈ C do
21:         C' ← {a}
22:         while C' ≠ ∅ do
23:           if W(C') + W(C \ C') > W(C) then
24:             W(C) ← W(C') + W(C \ C')
25:             B(C) ← C'
26:           end if{Evaluate subset}
27:           C' ← nextslyce(C', C, m)
28:         end while{Calculate w(C)}
29:       end for
30:     end if
31:   end for
32: end for
33: return bestcs(I)
```

Algorithm 4 *bestcs*(C)

```
if B(C) = ∅ then
  return {C}
else
  return bestcs(B(C)) ∪ bestcs(C \ B(C))
end if
```

when calculating $w(C)$, all possible splittings into feasible coalition subsets are considered, except $\{1, 4\}$ and $\{2, 3\}$. This means that $w(C)$ could possibly be less than $v(\{1, 4\})+v(\{2, 3\})$. However, the only coalition structure that contains $\{1, 4\}$ and $\{2, 3\}$ is $\{\{1, 4\}, \{2, 3\}, \{5\}\}$ and $w(I)$ is still greater than or equal to $w(\{2, 3\}) + w(\{1, 4, 5\})$ which is greater than or equal to $v(\{2, 3\}) + v(\{1, 4\}) + v(\{5\})$. So even though not every subset of every subset is considered, every coalition structure value is bounded by $w(I)$.

7. EMPIRICAL EVALUATION

In this section, we evaluate D-SlyCE and DyCE on a number of synergistic graph topologies. We benchmark D-SlyCE against DCVC, and DyCE against IDP and IBM's ILOG CPLEX (which solves the well known Integer Program formulation for the set partitioning problem posed by CFSS). Since our claim is that D-SlyCE and DyCE are particularly good in problems involving sparse graphs, we experiment

with a variety of graphs of different densities. While, in the case of D-SlyCE, as argued in Section 5, we expect the distribution of enumeration tasks among agents to depend on the degree of the graph, in DyCE, we expect the degree of the graph to affect the time to evaluate all feasible coalition structures and the cost of checking the size of individual coalitions (as in *nextslyce*).

In our experiments,[3] we focus on those topologies most commonly found in synergistic networks such as social networks, the internet, and peer-to-peer communication between emergency responders, as follows: (i) Scale-free graphs — a network generated according to a power law. We use the standard Barabási-Albert [1] preferential attachment generation model, with parameters $k = 1, 2, 3$. Thus, as the graph is constructed, new agents are attached to k existing agents such that each new agent is attached to existing agent j with probability $\frac{d_j}{\sum_j d_j}$ where d_j is the degree of agent j for all $j \in I$. (ii) Random trees – an acyclic graph rooted at a vertex to simulate hierarchical organisations. These graphs are constructed by attaching each new agent to a single randomly picked existing agent and (iii) Complete graphs — an edge exists between each pair of agents, and so all coalitions are feasible. This is not a sparse graph and is unlikely to arise as a social context for large numbers of agents, but we include it as a worst case scenario. We next elaborate on the experiment results for D-SlyCE and DyCE in turn.

7.1 Benchmarking D-SlyCE

In running D-SlyCE and DCVC on the graphs described above, we recorded the individual runtimes of each agent and computed the ratio between the mean runtime and the maximum for different numbers of agents. This is used to analyse the fairness of the computation distribution among the agents. Using our setup, given this, we are able to run DCVC for up to 40 agents in 2.6 hours. For complete graphs

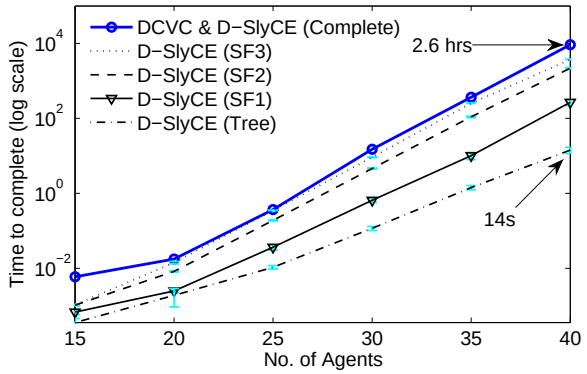


Figure 3: Runtimes for D-SlyCE (SFk = scale-free with parameter k).

[3]All our experiments are carried out on a 64 bit, quad-core PC with 12GB of RAM. In evaluating D-SlyCE, we repeated each experimental run 100 times (except for 40 agents where we ran only 20 times given the long runtimes) and for DyCE (on the more complex coalition structure generation problem), we repeated all the experiments 50 times (except for 28 agents onwards for trees/scale free parameter 1 and 23 agents onwards for the rest, which we repeated 20 times). In both sets of experiments, we recorded the mean, variance and 95% confidence intervals of the runtimes of the algorithms under study.

(see Figure 3 for runtime results), our algorithm matches the performance of DCVC, and, over sparse graphs (trees and scale free), outperforms DCVC for all numbers of agents. For trees and scale-free graphs, the worst case is 2.5 times faster (3700s compared to 9300s for a scale-free graph with $k = 3$ at 40 agents) and, in the best case, it is about 660 times faster (14s compared to 9300s for DCVC on a tree with 40 agents)! We note that in the case of random trees, D-SlyCE can evaluate all feasible coalitions for 40 agents within about 14 seconds and indeed D-SlyCE easily runs on trees of up 50 agents within 14 minutes! To date, no existing distributed coalition value calculation algorithm has been shown to have comparable performance. Also, note that as the density of the graph increases (i.e., as the number of connections per agent increases in k), the runtimes of the D-SlyCE algorithm increase as expected. This confirms that the density of the graph is a key determinant of the improvement that D-SlyCE makes over DCVC. Turning to the fairness of the computation distribution, as measured by the ratio of mean agent runtime against maximum runtime (such that 1 = equal computation distribution), the values obtained were (with SFk = scale-free with parameter k) — Trees: 0.09, SF1: 0.3, SF2: 0.6, SF3: 0.8, and Complete graph: 1 respectively. In the case of trees, a number of vertices that are well connected (e.g., high in the hierarchy) create disproportionately large task shares. However, in the increasingly dense scale-free graphs, all agents tend to get an increasingly fairer share of the computation as their relative degrees get closer (with the complete graph generating a perfect split of shares). When relating these results to the runtimes, however, we can see that in the case of random trees, the low overall computation time more than compensates for the unfair distribution.

7.2 Benchmarking DyCE

In this section, we describe two experiments (Exp1 and Exp2 respectively), one to evaluate how DyCE, IDP, and CPLEX compare in terms of runtime on the same instances and one where we evaluate the performance of DyCE as the graph density increases (see results in Figure 5). The latter is particularly important to consider since the number of feasible coalition structures (and calls to *nextslyce*) increases with graph density (see Algorithm 3). Thus, the gain from identifying feasible coalition structures is expected to tail off as the number of redundant coalition structures decreases (as graph degree increases).

Exp1: In terms of runtime, our results (see Figure 4) clearly show that DyCE outperforms IDP and CPLEX significantly on scale-free graphs and random trees. On complete graphs (by up to 7×10^4 times on trees for 30 agents compared to IDP), DyCE incurs an overhead that tails off (indicating a lower growth than that of the algorithm) and runs slower than IDP by a small amount. Due to long runtimes (and given that the trends are deterministic (i.e., growing in $O(3^n)$ for IDP) we extrapolated the results as follows: from 24 agents onwards for DyCE (Complete) and IDP, and from 29 onwards for DyCE (SF3) and 30 agents for DyCE (SF2). CPLEX, instead, is fast on small problems but surprisingly runs only up to 27 agents on trees (19 on SF3, 20 on SF2, 25 on SF1) — we do not plot these graphs for clarity). Moreover, we note that DyCE ran to completion within 5.3 minutes for 30 agents on trees.

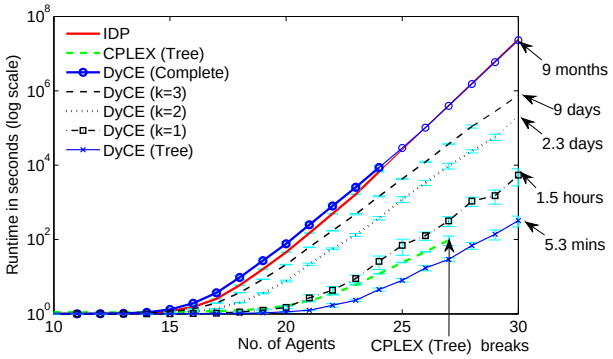


Figure 4: Runtimes for DyCE, IDP, and CPLEX.

Exp2: In this experiment, we evaluated DyCE with 20 agents on a scale-free graph with parameter $k \in \{1, 10\}$, where $k > 5$ results in a graph where each agent will be connected to more than half the number of agents (hence a dense graph). Given this, we note (from figure 5) that DyCE outperforms IDP for values of k up to 5, beyond which the graph is so dense that the coalition size checks in *nextslyce* become slightly more costly than the gain in avoiding infeasible coalitions which renders DyCE slightly slower on complete graphs (a fixed overhead which we believe could be improved through a faster implementation of *nextslyce*). Hence, it can be concluded that DyCE will work very well for most domains which have reasonably sparse synergies.

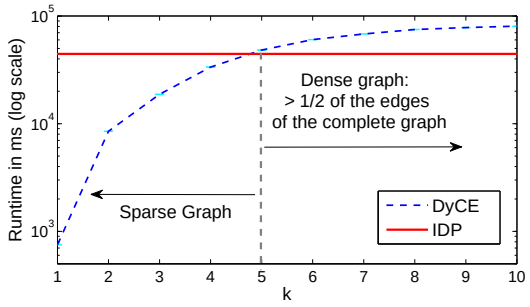


Figure 5: Runtime for $k \in \{1, 10\}$ in SFk ($n = 20$).

When taken together, our results allow us to say that D-SlyCE is the fastest distributed coalition enumeration algorithm over constraining graphs, and can help solve problems of more than 40 agents within reasonable time (in seconds, minutes, or hours depending on the graph). Moreover, we establish the benchmark for coalition structure generation algorithms in constraining graphs as DyCE can solve problems for up to 30 agents within minutes as compared to months for the state-of-the-art IDP.

8. CONCLUSIONS

In this paper we addressed the problem of coalition formation with sparse synergies where the set of feasible coalitions is constrained by the edges of a graph. Our aim was to see whether knowledge of the topology of an underlying social or organisational context graph could be used to speed up the task of coalition enumeration and structure generation.

We first developed the SlyCE algorithm and D-SlyCE to enumerate and evaluate all feasible coalitions over any graph in a distributed fashion such that agents end up with a fair share of computation. Theoretical results showed that (D-) SlyCE is correct, complete, and non-redundant. We then turned to the more challenging problem of coalition structure generation and proposed the DyCE algorithm to solve it using SlyCE as a building block.

Our empirical evaluation of D-SlyCE and DyCE, showed that they both outperformed the state-of-the-art algorithms by orders of magnitude (660 times for D-SlyCE and 7×10^4 times for DyCE), and for the first time, managed to enumerate coalition values for up to 50 agents in reasonable time, and find the optimal coalition structure for 30 agents within 5.3 minutes on random trees.

In general, our algorithms establish the benchmarks for many multi-agent applications where sparse synergies exist (e.g., decentralised coordination of sensors or emergency responders) where computing the optimal solution has, so far, not been possible due to the exponential time required to solve the coalition formation problems they generate. In future work, we aim to further improve DyCE by combining it with branch-and-bound techniques to further speed up the search and prune the synergistic graph to render the search feasible within reasonable time while providing quality guarantees on solutions returned. We also seek methods to automatically identify whether a problem has sparse synergies (i.e., based on graph degree) and thus help in the choice of the right algorithm to use.

9. REFERENCES

[1] R. Albert and A. L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74:47–97, 2002.

[2] G. Demange. On group stability in hierarchies and networks. *Journal of Political Economy*, 112(4):754–778, 2004.

[3] T. Michalak, J. Sroka, T. Rahwan, M. Wooldridge, P. Mcburney, and N. R. Jennings. A distributed algorithm for anytime coalition structure generation. In *Autonomous Agents And MultiAgent Systems (AAMAS 2010)*, pages 1007–1014, 2010.

[4] R. Myerson. Graphs and cooperation in games. *Mathematics of Operations Research*, pages 225–229, 1977.

[5] T. Rahwan and N. R. Jennings. An algorithm for distributing coalitional value calculations among cooperating agents. *AIJ*, 171(8-9):535–567, 2007.

[6] T. Rahwan and N. R. Jennings. Coalition structure generation: Dynamic programming meets anytime optimisation. In *Proc 23rd Conference on AI (AAAI)*, pages 156–161, 2008.

[7] T. Rahwan and N. R. Jennings. An improved dynamic programming algorithm for coalition structure generation. In *Proc 7th Int Conf on Autonomous Agents and Multi-Agent Systems*, pages 1417–1420, 2008.

[8] T. Rahwan, T. Michalak, E. Elkind, P. Faliszewski, J. Sroka, M. Wooldridge, and N. R. Jennings. Constrained coalition formation. In *The 25th Conference on Artificial Intelligence (AAAI)*, 2011.

[9] T. W. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *AIJ*, 111(1–2):209–238, 1999.

[10] O. Shehory and S. Kraus. Methods for task allocation via agent coalition formation. *AIJ*, 101(1-2):165–200, 1998.

Decentralised Channel Allocation and Information Sharing for Teams of Cooperative Agents

Sebastian Stein* ss2@ecs.soton.ac.uk
Simon A. Williamson† swilliamson@smu.edu.sg
Nicholas R. Jennings* nrj@ecs.soton.ac.uk

*University of Southampton, SO17 1BJ, Southampton, UK
†School of Information Systems, Singapore Management University, Singapore

ABSTRACT

In a wide range of emerging applications, from disaster management to intelligent sensor networks, teams of software agents can be deployed to effectively solve complex distributed problems. To achieve this, agents typically need to communicate locally sensed information to each other. However, in many settings, there are heavy constraints on the communication infrastructure, making it infeasible for every agent to broadcast all relevant information to everyone else. To address this challenge, we investigate how agents can make good local decisions about what information to send to a set of communication channels with limited bandwidths such that the overall system utility is maximised. Specifically, to solve this problem efficiently in large-scale systems with hundreds or thousands of agents, we develop a novel decentralised algorithm. This combines multi-agent learning techniques with fast decision-theoretic reasoning mechanisms that predict the impact a single agent has on the entire system. We show empirically that our algorithm consistently achieves 85% of a hypothetical centralised optimal strategy with full information, and that it significantly outperforms a number of baseline benchmarks (by up to 600%).

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI—*multi-agent systems*

General Terms

Algorithms

Keywords

teamwork, multi-agent learning, communication

1. INTRODUCTION

It is envisaged that teams of heterogeneous software agents will be increasingly used to tackle complex real-world problems. These include intelligent sensor networks, autonomous vehicles that explore hostile environments and portable devices that provide emergency responders with situational awareness during a disaster situation. In all these multi-agent systems, and many others besides, coordination is typically achieved through communication between the agents, who share their beliefs about the state of the problem so that together they can find a better, coordinated response.

However, communication infrastructures in real-world problems often have considerable constraints. As dedicated high-bandwidth networks are costly to develop and deploy, or may simply be unavailable in an emergency situation, agents often need to rely on very limited existing communication means. These may include using mobile ad hoc networks [3], power line communications [13] or cognitive radio [1], where agents compete to utilise the spare capacity of other communication networks via spectrum sharing and channel allocation [11]. Now, using any of these constrained media introduces a new coordination challenge: how to best utilise the available capacity to communicate information effectively across agent teams.

To date, research on such systems has largely concentrated on finding a fair division of bandwidth between competing, self-interested agents in a decentralised setting. For example, there are protocols such as FDMA (Frequency Division Multiple Access) [1] and approaches that model the problem as a multi-armed bandit [12, 2] or as a congestion game [10]. However, all these solutions assume identical information needs and that each agent is only interested in maximising its own bandwidth.

Hence, existing work neglects the fact that a constrained communication system may be used by cooperative teams of agents to solve a joint problem whose global utility is not maximised by giving each of them fair access. To exemplify this challenge, we ground our work on a disaster management scenario where teams of ambulance, fire brigade and police agents must respond to an earthquake in an urban area (such as seen in the RoboCupRescue[1] competition [8]). Here, the overall joint problem is to contain the disaster and minimise damage to civilians and property. However, the individual capabilities of agents are very specific — ambulance agents rescue civilians, police agents clear roads and fire brigades control fires. Therefore, while agents may discover any type of information (e.g., an ambulance may discover a new fire), their individual information needs are highly heterogeneous (e.g., fire brigades are mostly interested in detailed information about the location and severity of fires), and to solve the overall problem effectively, relevant information needs to reach the right agents.

Now, in this setting, agents can communicate such information using a limited set of channels, which represent different parts of the radio spectrum or even entirely different communication media. However, communication on these channels is severely constrained, both in the number of channels an individual agent can access simultaneously (due to technological or cognitive constraints) and in the available bandwidth on each channel. Moreover, these constraints typically preclude the use of explicit or centralised co-

[1]Another application is distributed sensor optimisation, where different types of sensors for tracking environmental phenomena are coordinated, whilst trying to share an underlying restricted communication infrastructure. Also, to reduce deployment costs, ubiquitous computing often utilises existing communication media, which are shared between applications, e.g., in a smart house.

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

ordination, due to time constraints and because coordination messages would themselves congest the channels. Examples of such coordination approaches include decentralised task allocation models such as Max-Sum [7], reward shaping [15] or auctions [9], which consider how to optimally allocate tasks and distribute disparate beliefs. However, these models often make heavily restrictive assumptions about the communication infrastructure and network topology, which do not hold in the settings we consider (such as assuming that neighbours in a network can communicate with each other at zero cost).

Another option is to solve the communication problem optimally, by casting it as a decentralised POMDP [14]. Here, both centralised and decentralised algorithms can be used, which analyse the future impact of sending a specific message on a given channel [4]. However, this approach is computationally intractable [5] and not appropriate for a scalable solution in systems with hundreds of agents.

To address this setting, we propose a new decentralised approach that converges to a good overall channel allocation, which, unlike previous work, considers the asymmetric information needs of agents. Specifically, it generally allocates agents that are interested in certain types of information to specific channels (such that, e.g., all fire brigades use a specific channel to share information about fires). However, in doing this, our approach considers subtle synergies between different agents, potentially placing several heterogeneous types of agents on the same channel if they share common information needs. It also takes into account bandwidth availability and team sizes to select appropriate channels, and it deals flexibly with low channel availability by sometimes allowing even agents with no common information needs to share the same channels. Crucially, this behaviour is based on principled calculations that predict the overall impact of a channel allocation on the global system utility, and it requires no a priori coordination or channel assignment. Furthermore, agents in our approach continuously monitor and learn the value of information locally to decide when to break from a given allocation, either temporarily (to pass on information to other types of agents) or permanently (to find a better overall allocation).

In addressing this problem, we make several key contributions. First, we present a novel decentralised channel allocation problem and show that solving it is inherently hard. Then, we develop our new decentralised algorithm that uses local learning and decision rules to solve it. Finally, we show empirically that this converges to a good solution that is typically within 85% of a hypothetical centralised optimal approach and outperforms a number of standard benchmarks (achieving a 6-fold improvement in some cases).

The remainder of this paper is structured as follows. In Section 2, we formalise the channel allocation problem we solve in this paper and then discuss how this problem could be solved in a centralised manner in Section 3. This discussion then informs our decentralised learning solution, which we outline in Section 4. In Section 5, we evaluate our approach experimentally and conclude in Section 6.

2. CHANNEL ALLOCATION PROBLEM

We begin by formally outlining the channel allocation problem (CAP) we address in this paper. Specifically, we consider a system that consists of heterogeneous *agents* that need to share *facts* about the world with each other. These facts constitute key items of information about the state of the world that help the agents in solving their joint problem. Generally, these facts could include sensor readings, locations of other agents and new tasks that have been discovered. Normally, we assume these facts are known with complete certainty (but they could also represent probabilistic information that improves the agents' decisions). Critically, not all facts are equally important and some may only be of interest to a subset of agents.

More formally, $\mathcal{A} = \{a_1, a_2, \ldots, a_A\}$ denotes the set of agents, each of which has a type given by $m : \mathcal{A} \rightarrow \mathcal{S}$, where $\mathcal{S} = \{s_1, s_2, \ldots, s_S\}$ is the set of all types (we will sometimes refer to the agents of a specific type as a *team*). Furthermore, $\mathcal{F} = \{f_1, f_2, \ldots\}$ is the set of all possible facts, with each fact having a deadline $d : \mathcal{F} \rightarrow \mathbb{N}_0^+$, which refers to a specific time step (we assume discrete time steps), after which the fact is no longer relevant to any agent. For example, such a deadline could represent when a building on fire burns out or when an injured civilian can no longer be saved.

Now, in our model, agents derive an explicit reward for knowing about facts. This describes the relative importance of those facts and captures the intrinsic value that knowledge of them adds to the problem-solving ability of an individual agent. We assume these rewards are known a priori, representing the domain knowledge of the system designer, although in practice, they could be probabilistic estimates rather than deterministic values. In more detail, $r : (\mathcal{F} \times \mathcal{S}) \rightarrow \mathbb{R}^+$ describes the reward per time step that an agent of a particular type generates for knowing a specific fact (this function is available to all agents, e.g., a fire brigade is aware of the value of a civilian position to an ambulance). Thus, the total reward an agent generates per time step is the sum of the rewards (for its specific type) of all facts that it knows about and that have not exceeded their deadlines yet. Similarly, the overall reward generated in the system is the sum of all the agents' individual rewards, and this is the key objective we seek to maximise in our work.

We assume that new facts are discovered gradually over time by individual agents at the beginning of each time step (according to some random process), and they immediately start to generate rewards. However, to maximise rewards, agents can communicate the facts they know about on a set of highly constrained communication channels, $\mathcal{C} = \{c_1, c_2, \ldots, c_C\}$. Importantly, due to cognitive or technological constraints, agents can only use a limited number of channels each time step, as given by $l : \mathcal{S} \rightarrow \mathbb{N}_0^+$. When an agent decides to use a channel, it can post a single fact on that channel and listen to the facts that other agents have posted on that channel. Each channel has a finite bandwidth, $c : \mathcal{C} \rightarrow \mathbb{N}_0^+$, the number of facts that the channel can carry in a single time step. Any surplus facts are discarded uniformly at random.

Given this, the choice for the agents is which channels to subscribe to and which facts to send on these channels. This is done in two steps — first, all agents simultaneously choose the subset of channels they wish to subscribe to (up to their channel limit l and, critically, without knowledge of the others' choices), they then discover who else subscribed to the chosen channels, and finally all agents choose simultaneously what facts, if any, to post on those channels. At the end of the time step, agents are informed of all facts that were successfully posted on their subscribed channels and these now become shared knowledge among all subscribers (starting to generate rewards from the next time step).

Now, making these two decisions — first, deciding which channels to subscribe to, and, second, which facts to post on the channels — comprise the key channel allocation problem we consider in the paper. Its difficulty lies in the fact that agents cannot coordinate a priori on their choice of channels (and may therefore subscribe to channels that will contain no interesting facts) and because they do not know what facts others may post to the channel (and may therefore post low-value facts that displace other more valuable ones). Our overall aim here is to solve this problem in a decentralised manner with agents that use only local knowledge. However, to gain a better insight into what a good solution looks

like, we will first discuss a (hypothetical) centralised solution in the next section.

3. CENTRALISED SOLUTION

In this section, we present two centralised solutions for the problem of maximising the overall reward during channel allocation — one is optimal, but tractable only in small problems and the other is locally optimal and scales to larger problems. Although centralised solutions are infeasible in realistic settings, defining these provides us with useful benchmarks to compare our approaches against. Specifically, they constitute upper bounds for the performance of decentralised approaches, as they assume full information and control over all agents. Furthermore, examining the optimal centralised solution will guide our decentralised approach.

Now, as solving the CAP to maximise long-term rewards is intractable due to the huge search space (as in a decentralised POMDP), we here (and in our decentralised approaches) concentrate on myopic solutions. These consider only the actions available in the current time step and do not plan ahead. In small environments, where a long-term optimal can be found, we observed that the myopic performs close to this, as there is often little benefit in planning ahead and it is usually optimal to send facts that immediately generate large rewards.

However, we first show the complexity of the optimal solution.

3.1 Complexity Result

We will show that even the myopic version of the centralised CAP is NP-complete.

DEFINITION 1 (MYOPIC CENTRALISED CAP (MCCAP)). *Given the model in Section 2 and the agents' current beliefs (here, we let B_i denote agent i's belief, i.e., the facts that it is aware of), are the agents able to generate a total reward of at least x, for a given $x \in \mathbb{R}$, assuming that no more facts are generated or communicated in future time steps?*

First, recall the well-known NP-complete Hitting Set problem:

DEFINITION 2 (HITTING SET). *Given a collection S' of subsets of some universe U and a constant $k \in \mathbb{N}$, is there a subset $X \subseteq U$, such that X intersects every element of S' and $|X| \leq k$?*

THEOREM 1. *MCCAP is NP-complete.*

PROOF. We need to show that MCCAP is both in NP and also NP-hard. The first is straight-forward — given a solution of which channels each agent should subscribe to and what facts to send, we can calculate the reward generated by all agents in subsequent time steps in linear time and verify this is at least x.

To show that MCCAP is NP-hard we show that any instance of HITTING SET can be reduced in polynomial time to an instance of MCCAP. To do this, define $\mathcal{F} = U$, such that every fact corresponds to an element in U and set its deadline to 1, $d(f_j) = 1$ (assuming the current time is 0). Now, for each element $I \in S'$, create an agent a_I with an empty belief ($B_I = \emptyset$) and a unique type for that agent, s_I, i.e., $m(a_I) = s_I$. Limit the agent to subscribing only to one channel, $l(s_I) = 1$. Then, for each element $i \in I$, set the reward for a_I knowing the corresponding fact to 1, while all other rewards for that agent are 0, i.e., $r(f_i, s_I) = 1$ if $i \in I$, otherwise $r(f_i, s_I) = 0$. Finally, define an agent a_0 with type s_0 that knows all facts (with belief $B_0 = \mathcal{F}$), but set all its rewards to 0 ($r(f_i, s_0) = 0$ for all i). Set its subscription limit to k, i.e., $l(s_0) = k$. Finally, there are k channels, $\mathcal{C} = \{c_1, c_2, \ldots, c_k\}$, each with a bandwidth of 1. We now set the value threshold to $x = |S'|$.

Given this transformation, which can be performed in polynomial time in the size of the original instance, the solution of the original HITTING SET instance is *yes*, if and only if the solution to the new MCCAP instance is also *yes*. This is because each of the agents corresponding to the elements in S' generates a reward of 1 if and only if a fact corresponding to an element of its associated set $I \in S'$ is placed on one of the k channels (no more than 1 can be generated by each agent due to their subscription limits and the channel bandwidth constraints). Since at most k facts can be placed on the channels, all of these agents generate a reward (i.e., the overall value generated is $|S'|$), if and only if there is a subset of $X \subseteq U$ with $|X| \leq k$, such that it intersects all elements in S'. □

Given this, it is trivial to show that the non-myopic version of CAP is NP-hard, as the same transformation can be applied. This indicates that a centralised myopic solution is generally intractable, especially for larger problems. For this reason, we next present two algorithms: an optimal one that can be used in smaller settings and a suboptimal one that scales to larger problems. This allows us to see how close our decentralised algorithms are to optimal in small problems, and also have a good idea in larger problems.

3.2 Centralised Myopic Optimal

In the centralised myopic optimal solution (*OPTIMAL*), at every timestep x, a centralised authority gathers information about the facts held by each agent and solves the MCCAP instance optimally before distributing the allocation and facts to send to each agent. We achieve this by formulating MCCAP as an integer linear program and solving it using ILOG CPLEX.

Whilst this algorithm provides the optimal myopic solution, it is far from scalable. Specifically, even when looking for a myopic solution, the search space is doubly exponential in the number of agents A and the number of channels that can be subscribed to l. Next, we address the scalability of this solution.

3.3 Centralised Myopic Local Search

In the centralised myopic local search solution (*CMLS*), the same centralised authority gathers all information and distributes an allocation as for *OPTIMAL*. However, rather than considering the entire search space as in *OPTIMAL*, we start with a random solution and then carry out local improvements.

More specifically, given an initial allocation where agents randomly send facts to channels, we apply the single best deviation that an agent could perform (by switching or removing a fact to send, switching a subscribed channel or both). Then, we iteratively apply such deviations until a local optimum is reached and no more deviations yield a higher utility. The resulting allocation is then selected as the overall solution.

Clearly, this algorithm is suboptimal since it only allows a finite step size in the improvement phase. If this was relaxed, then combinations of new partial allocations could be considered in order to find a global optimum. However, the reduction in the state space used here does result in a substantially more scalable algorithm which is still guaranteed to find a local optimum for the myopic case. That said, the algorithms presented in this section are centralised and also compute solutions over all agents, facts and channels. However, it is useful to consider what an optimal solution to the channel allocation problem looks like, in order to inform our decentralised approach, as we do next.

3.4 Analysis

By examining actual channel allocation decisions made by the centralised solution, we can make several general observations about

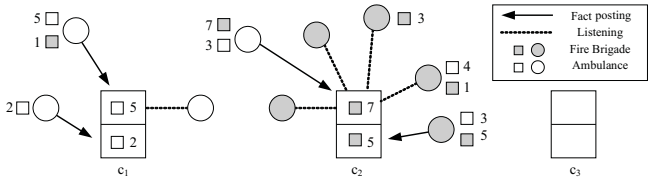


Figure 1: Example optimal channel allocation.

desirable behaviour in our problem domain. First, we note that generally it is beneficial for agents of the same type to use a dedicated channel for facts that they are interested in. That way, the overall reward generated by a posted fact is maximised as all interested agents receive it immediately, and there is typically no incentive for such a group to split up and use several channels. Conversely, it is usually not beneficial for agents of different types to share the same channel for their respective facts (unless the posted facts are relevant to both types), as this reduces the effective bandwidth of each team. Thus, our decentralised algorithm will dynamically designate channels to certain types and post only relevant facts to these.

Second, however, it is clearly suboptimal to restrict agents to posting and listening only to their designated channels. As each agent may find out about facts that are relevant to agents of other types, they need to occasionally communicate these by posting them to other channels (e.g., when an ambulance discovers a fire). However, this also means that the sending agent generates no additional reward by listening to more relevant facts on its own channel. For this reason, our algorithm will solve a local decision problem that explicitly balances the reward that the agent expects to generate by staying on its own channel with that generated by communicating a fact for a different agent type.

This general decision-making behaviour is reflected by the actions of the hypothetical centralised optimal strategy. To illustrate this, Figure 1 shows an optimal decision in an example scenario from RoboCupRescue. Note that this is the best allocation the agents could choose if they could perfectly coordinate their actions and had full knowledge of each others' available facts. Clearly, this is unrealistic, but we use it to illustrate what a good allocation looks like. In this example,[2] there are two types of agents, fire brigades and ambulances (shown as shaded and white circles, respectively). These can only subscribe to one channel each, but they have information about a variety of newly discovered facts that they can communicate to the others — these comprise information about both injured civilians (white squares) and fires (shaded squares). Ambulances are interested in injured civilians, while fire brigades derive rewards from finding out about fires (all rewards are noted in the figure). There are three channels with a bandwidth of two each.

In the optimal channel allocation shown here, we first note that channels are used exclusively for a particular type of fact (c_1 is used for facts about civilians and c_2 is used for fire), and most of the agents only listen to the channels for their respective type. The only exception to this is the ambulance in the centre (with one fire fact worth 7 and a civilian fact worth 3), which posts information about a particularly critical fire to the channel of the fire brigades. Although this means that this ambulance does not gain any reward itself from hearing about civilians, the value of its information to the fire brigades is high enough to warrant this decision. We note also that the third channel, c_3, is completely unused, because moving any of the agents and posting further facts there does not increase the overall rewards generated.

We will use these observations in the construction of a decentralised algorithm which avoids the complexity issues of the centralised approaches, but still performs well in comparison.

[2]An equivalent example from sensor networks sees two types of sensor: a ground based static sensor for detecting troops and UAVs for tracking faster-moving vehicles. These may detect targets for each other but would not take any direct action, instead communicating them to the respective agent. In ubiquitous computing, energy management devices need to communicate information about power usage with each other; however, the security system may need to track a threat and switch on high power devices over that same channel.

Algorithm 1 Basic Decentralised Learning Algorithm

1: **while** *true* **do**
2: Observe new facts and update fact beliefs.
3: Choose channels and facts to post using local beliefs.
4: Subscribe to channels and post facts.
5: Update channel beliefs.

4. DECENTRALISED SOLUTION

In this section, we develop a decentralised learning algorithm to solve this problem in realistic settings. With this, agents use only local information and previous observations to decide which fact to send on which channel. More specifically, we assume agents know only the facts they have discovered themselves and those they have heard on channels, but they do not know what facts are known by others. They also know the channels and their characteristics, and they know what other agents are present in the system and their types (although our algorithm can easily be extended to discover this at run-time).

As we argued in Section 1, this problem can be solved optimally by casting it as a decentralised POMDP. However, that methodology is intractable for even small agent teams. Further to this, that solution still requires a complex coordination mechanism (or communication) to execute the decentralised policies and full knowledge of the fact generating function in order to obtain the policies in the first place. Consequently, we look towards reinforcement learning of the underlying problem, together with a simple coordination mechanism, as an efficient means to finding a decentralised solution. Now, this section will first describe the overall approach we take in our solution, followed by its individual components and finally the overall algorithm is presented.

4.1 High-Level Approach

Our problem requires that the agents find a common agreement on how to divide the channels into their respective types, and then, agents need to decide when they should send facts (or listen) to their own channel and when they should communicate facts to other types. We address these two challenges separately, focusing first on the channel division and then on the fact communication problem.

Before considering the details, Algorithm 1 shows the high-level approach used in our decentralised solution. To make good decisions, our algorithm maintains beliefs about the system, including *fact* beliefs about the characteristics of the fact generation process in the system, and *channel* beliefs, which include historical frequencies over what types of agents have been observed on the various channels and some information about the value of facts that have been posted to the channels in the past. These beliefs are updated regularly at every time step based on new facts that are observed (line 2) as well as the agents and facts that are seen on channels (line 5).

The key decision about what channels to subscribe to and what facts to post is made in line 3. This is clearly critical, but it is important here to note that this can depend only on the agent's local beliefs, without knowing the beliefs of other agents or their future

decisions. However, these local beliefs (in particular the channel beliefs) are shaped by the past actions of other agents in the system. As such, our algorithm constantly *learns* and *adapts* to the actions of others, allowing it to respond better in the future. This is very similar to fictitious play [6], which has been used successfully in more competitive settings, such as congestions games. However, although our setting is cooperative in nature, fictitious play has several desirable properties — it is known to converge to Nash equilibria, and it represents a simple, tractable learning technique (rather than resorting to complex optimal approaches, such as POMDPs). This makes it suitable for our problem.

As part of making the decision about what channels to subscribe to and what facts to post, an agent needs to first know what teams are assigned to what channels. We will consider this question in Section 4.2. Second, given that it has this channel allocation, it then needs to decide what facts to post (if any) and on what channels. We treat this problem in Section 4.3. Finally, we present an overall algorithm in Section 4.4.

4.2 Channel Division

Dividing the channels effectively between the agent types is a central part of our algorithm. The aim of this is to find a function $division : \mathcal{S} \rightarrow \mathbb{P}(\mathcal{C})$, which maps each agent type to the set of channels used by that type (note these can overlap). All agents need to agree on this and the mapping needs to also consider the bandwidth of channels (it may be more beneficial to share a single channel with a large bandwidth between several types than have one type use a very small channel).

To quickly find a consensus for the *division* function, the agents treat their own type and other types differently. Specifically, they assign other types simply to the channels that have the highest participation by those agents, based on previous observations made when subscribing to channels, up to the channel limit of that type. For its own channel assignment, each type first designates a leader, who decides on the best channels for its type.[3]

The leader's decision is based on probabilistic knowledge about arrivals of new facts (its fact beliefs), which it uses to calculate the expected utility of choosing a particular channel. In more detail, we denote by $\overline{u}(\mathcal{S}', s_i)$, with $\mathcal{S}' \subseteq \mathcal{S}$ and $s_i \in \mathcal{S}$, the total reward an agent of type s_i expects to gain from a randomly chosen fact, given that the fact has a non-zero reward for at least one of the members of $\mathcal{S}'$. With this, the expected utility generated by a particular set of agent types $\mathcal{S}'$ using channel c_i is estimated as:

$$\overline{r}(\mathcal{S}', c_i) = c(c_i) \cdot \sum_{s_j \in \mathcal{S}'} \overline{u}(\mathcal{S}', s_j) \cdot \left(\text{size}(s_j) - \frac{\text{size}(s_j)}{\text{total}(\mathcal{S}')} \right), \quad (1)$$

where $\text{size}(s_j)$ is the total number of agents of type s_j and $\text{total}(S') = \sum_{s_k \in \mathcal{S}'} \text{size}(s_k)$. As such, it is the expected reward all agents on channel c_i expect to generate through communication per time step, given that the channel is always fully utilised for sending new facts, that agents choose to send facts randomly from among those that are relevant to at least one other agent on the channel and that all agents in the team subscribe to the channel. These are simplifying assumptions, but they serve to allow an agent to quickly compare the relative merit of choosing a particular channel over another. In doing so, the calculation considers the relative sizes of a team on a channel (as all agents but the sender benefit from a posted fact), the bandwidth of the channel, as well as taking into account the relative frequencies of facts, e.g., when facts for a particular type are more frequent than those for another type (this is intrinsically part of $\overline{u}(\mathcal{S}', s_j)$).

Given this, the leader then uses Equation 1 to evaluate the overall impact of choosing a particular channel for its type (keeping the allocations of other types constant, as given by the *division* function), and greedily chooses the channels with the highest respective utilities, up to the channel limit for its type. Its other team members, in turn, also adopt these chosen channels (if necessary by searching for the channels the leader subscribes to, as will be explained later). Assigning a single leader agent in this way allows the team to converge quickly to a set of channels, which can be easily recomputed at any time, e.g., to take into account dynamically changing reward distributions, team sizes or available channels.

4.3 Fact Communication

So far, we have described how agents reach consensus about a consistent channel division between the agent types. However, the agents only derive value from actually sending facts to channels. As we argued in Section 3.4, agents will generally send facts only on their own channels, unless they have a particularly valuable fact for a different type. To determine which facts to send to which channels, we use a decision-theoretic approach and estimate the expected total reward that an agent a_l would contribute to the system by sending a given fact f_i at time t to channel c_j:

$$\mathbb{E}(r_{\text{send}}(a_l, f_i, t, c_j)) = \sum_{s_k \in \text{onChannel}(c_j)} \text{size}_{-l}(s_k) \cdot (r(f_i, s_k) \cdot (d(f_i) - t) - (1 - \text{total}_{-l}(c_j)^{-1}) \cdot \overline{s}(s_k)), \quad (2)$$

where $\text{onChannel}(c_j)$ is the set of agent types that are mapped to channel c_j by the *division* function, $\text{size}_{-l}(s_k)$ is the number of agents of type s_k excluding a_l, $\text{total}_{-l}(c_j)$ is the total number of agents of the types given by $\text{onChannel}(c_j)$ excluding a_l and $\overline{s}(s_k)$ is the reward that a single bandwidth unit on a channel for the given type s_k is expected to generate (again, based on previous observations, and, to avoid bias, excluding the facts the agent previously posted itself). Thus, this equation is positive if the fact improves on what is otherwise expected to be posted on the bandwidth unit it would occupy.

As this assumes that posted facts are new to all agents on the channel, we allow agents to only post facts they themselves have discovered and that have not been sent successfully to this channel (or other channels occupied by the same types) before — this greatly simplifies the decision problem, as agents do not need to keep track of the belief states of other agents.

Apart from $\mathbb{E}(r_{\text{send}}(f_i, t, c_j))$, agent a_l expects to benefit from listening when posting to a particular channel c_j (if this is one of its own channels). We denote this expected benefit as $\mathbb{E}(r_{\text{listen}}(a_l, c_j))$ and note that it is 0 when the channel is not one of its own (i.e., $m(a_l) \notin \text{onChannel}(c_j)$), otherwise it is $\overline{s}(m(a_l)) \cdot (l(c_j) - 1)$.

Thus, we can obtain an overall valuation for an agent a_l of sending a fact f_i at time t to channel c_j:

$$\mathbb{E}(r_{\text{overall}}(a_l, f_i, t, c_j) = \mathbb{E}(r_{\text{send}}(a_l, f_i, t, c_j)) + \mathbb{E}(r_{\text{listen}}(a_l, c_j)) \quad (3)$$

Using Equation 3, the agent can now greedily choose the best facts to post, or stay silent and listen to its own channels when this is more beneficial (the utility of this is $\overline{s}(m(a_l)) \cdot l(c_j)$ — slightly higher than $\mathbb{E}(r_{\text{listen}}(a_l, c_j))$ because the agent potentially hears one additional fact).

[3] This can be achieved without explicit communication, using, e.g., unique identifiers, such as IP or MAC addresses of the agents encountered so far. Throughout this section, we will assume that an agent has a priori knowledge of all other team members and their identifiers, but even when this is not available, a simple adaptive leader election protocol based only on team members observed so far could be designed. We leave this, and the re-assignment of missing leaders, to future work.

4.4 Overall Algorithm

Our approaches for choosing a consistent channel allocation and deciding what facts to post on what channels constitute the core of our decentralised algorithm. In addition to these decision-making procedures, however, it is important for the agents to sometimes choose actions that allow them to better update their beliefs, rather than simply maximise their expected rewards through posting facts. This is because our algorithm relies on some statistical knowledge that is learnt at run-time, but when this is inaccurate, an agent may lack the incentive to further subscribe to the affected channels and update its knowledge. This exploration/exploitation tradeoff is common in learning problems, and we adopt an approach that is often employed there: ε-greedy. More specifically, with a small probability[4] $\varepsilon = 0.01$, agents *randomly* pick a subset of channels to subscribe to, rather than considering all possible channels. This ensures both that the *division* function is updated when other teams move channels, and, similarly, that $\bar{s}$ is learnt.

Given this, Algorithm 2 summarises the overall decision-making mechanism each agent follows. In brief, each agent starts a time step by observing its environment and gathering new facts about the world (line 5). These are used to update the agent's local $\bar{u}$ function (line 6). Next, the agent calls a procedure depending on whether it is a leader or not (line 9).

As a leader, it first re-evaluates the current channel division with a small probability, $\varphi = 0.05$. Here, the FINDBESTDIVISION procedure (line 16) finds the best channels to use for the agent's own type, given the current *division* for all other types (as described previously), and returns this only if it is at least $\delta = 5\%$ better than the current choice. Note that φ and δ are included here to prevent the agent from switching channel divisions too quickly. If no exploration takes place, and the leader has not subscribed to its own channels for at least $maxAbsence = 5$ time steps, it is forced to only consider its own type's channels (line 18), which allows other agents to easily detect when the leader has chosen to change channels (hence, a longer absence indicates that the leader has changed channels). If this is not the case, the leader chooses random channels with probability ε, where RANDOMCHANNELS returns a set of random channels of size equal to the agent's subscription limit (line 20).

As a follower, if the leader has not been observed for more than *maxAbsence* time steps, the agent starts searching, where SEARCHCHANNELS returns channels that have not been visited recently (line 23). Otherwise, the follower also chooses random channels with probability ε (line 25).

Next, the agent chooses the best facts to post to the eligible set of *channels* (line 10). This is done using the procedure described in the previous section. The agent then follows this, first subscribing to channels and then posting facts (lines 11 and 12). Finally, it uses information observed on its subscribed channels (i.e., participating agents and posted facts) to update a number of statistics (line 13).

Concluding this section, as we noted earlier, our algorithm can be seen as a form of fictitious play, i.e., each agent effectively plays the best response to the other agents' actions (as learnt by the *division* and $\bar{s}$ functions). This allows our strategy to adapt to a dynamic environment. For example, when only a few, low-value facts are sent to a particular channel (as indicated by a low average reward per bandwidth, $\bar{s}$), agents decrease their threshold for sending facts, but when congestion is high, only very valuable facts are sent.

[4]We stress our algorithm does not depend on the exact choice of this parameter and others mentioned in this section. However, we list the parameters used in our implementation for completeness.

Algorithm 2 Decentralised CAP Algorithm (DecCAP)

```
1:  a_i ← myself                                              ▷ Agent executing this
2:  leaderAbsent ← ∞                                          ▷ When was the leader last seen?
3:  ownFacts ← ∅                                              ▷ Facts discovered by this agent
4:  while true do                                             ▷ For each time step
5:      newFacts ← OBSERVEENVIRONMENT()                       ▷ Gather new facts
6:      ū ← UPDATEFACTSTATS(newFacts)                         ▷ Update fact statistics
7:      ownFacts ← ownFacts ∪ newFacts
8:      channels ← C                                          ▷ Channels to consider
9:      LEADERLOGIC() / FOLLOWERLOGIC()                       ▷ Depends on whether leader
10:     decision ← CHOOSEBESTFACTS(channels, ownFacts)        ▷ Facts to post
11:     SUBSCRIBE(decision)                                   ▷ Subscribe to chosen channels
12:     heardFacts ← POSTFACTS(decision)                      ▷ Post facts
13:     division, leaderAbsent, s̄ ← UPDATECHANNELSTATS(heardFacts)
14: procedure LEADERLOGIC( )                                  ▷ Leader's decision-making logic
15:     if Random(0,1) ≤ φ then                               ▷ Explore new channel division?
16:         division ← FINDBESTDIVISION(division, δ)
17:     else if leaderAbsent ≥ maxAbsence then                ▷ Stick to own channels?
18:         channels ← division(m(a_i))
19:     else if Random(0,1) ≤ ε then                          ▷ Random exploration?
20:         channels ← RANDOMCHANNELS()
21: procedure FOLLOWERLOGIC( )                                ▷ Follower's decision-making logic
22:     if leaderAbsent > maxAbsence then                     ▷ Search for leader?
23:         channels ← SEARCHCHANNELS()
24:     else if Random(0,1) ≤ ε then                          ▷ Random exploration?
25:         channels ← RANDOMCHANNELS()
```

5. EMPIRICAL RESULTS

In this section, we comprehensively evaluate the performance of our algorithm, *DecCAP*, against the interesting space of problems captured by our model. To measure its performance, we compare it to a number of benchmarks:

- *OPTIMAL/CMLS* is the centralised optimal[5] algorithm from Section 3. As such, it is clearly not realistic in practice, but rather serves as an *upper bound*.
- *RANDOM* subscribes to random channels placing a single random fact from its current beliefs on each channel.
- *BEST-FACT* subscribes to random channels and then places the fact that promises to generate the highest reward on each channel. This is based on the agent's local beliefs about what facts are already known by others.
- *EPSILON-GREEDY(ε)* generally subscribes to the channel that has generated the highest overall average reward for that agent (based on past observations), but with probability ε, it picks a random channel instead for exploration. When subscribed, it behaves as the *BEST-FACT* strategy. As such, it represents a common solution approach to work that models the channel allocation problem as a multi-armed bandit [2].

In the following, we restrict our analysis to a problem setting from the recent RoboCupRescue competition (since this was created to be a taxing problem with all of the challenges discussed earlier) and then explore some interesting parameters within this domain. Specifically, we first consider the standard setting from RoboCupRescue with three types of agents: ambulances, police and fire brigades. These are interested in three types of information: new civilians, road blocks and fires. We assume facts are discovered randomly, such that each agent discovers one new fact on average every four time steps (using a Poisson distribution), each fact is of a random type, has a reward drawn uniformly at random from $[0,1]$, and a deadline drawn uniformly at random from

[5]Due to the complexity of this, we use myopic optimality here, and, for more than 10 agents, we replace the *OPTIMAL* by the greedy *CMLS*, which achieves the *same* performance as *OPTIMAL* on the smaller settings.

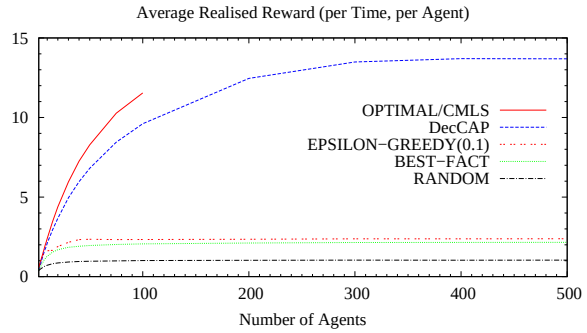


Figure 2: Performance as the number of agents is increased.

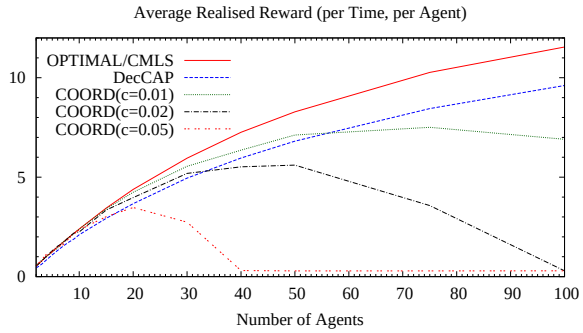


Figure 3: Performance with explicit coordination.

$\{2, 3, \ldots, 10\}$ (to represent buildings that burn out or civilians that die). We also consider a highly constrained communication infrastructure with five channels that can contain only two facts each.

Given this setting, we are first interested in establishing the relative performance of our approach to the benchmarks and to evaluate whether it scales to large systems.

5.1 Basic Benchmarks

To first establish how *DecCAP* performs in increasingly large settings, Figure 2 shows the results[6] with increasing numbers of agents. It is clear here that *DecCAP* significantly outperforms the two baseline benchmarks, *RANDOM* and *BEST-FACT*, improving on them by up to 600%, as it is able to select a good channel allocation and communicate the best facts to the right agents. The learning approach *EPSILON-GREEDY* is also outperformed by *DecCAP* (we only plot $\varepsilon = 0.1$ here as one of the best performing parameter choices). This is because the former quickly converges to a local optimum, in which all agents communicate on the same channel.

Finally, we observe that *DecCAP* also consistently achieves 85% or more of *OPTIMAL* and *CMLS*. This is a significant result, given that those assume full information and complete control over all agents' actions. Finally, we note that we could only run *CMLS* up to 100 agents, beyond which it became exceedingly slow, while *DecCAP* still made fast decisions in less than 0.5 ms per agent and time step with 1000 agents, using a Java implementation on an Intel 2.2GHz laptop (not shown on the graph for readability).

5.2 Explicit Coordination

While we argue that the *OPTIMAL/CMLS* centralised approach is unrealistic in most settings, it could be achieved in practice through the use of explicit coordination between agents. In order to do this, agents need to exchange coordination messages, which places an additional burden on the communication infrastructure and thereby reduces the effective bandwidth that can be used to exchange facts.

To capture this class of coordination approaches, exemplified by the Max-Sum algorithm [7] or auctions for resource allocation [9], we define a new benchmark, *COORD(c)*, which behaves as the centralised approach, but incurs a small communication cost of c per agent in the system. Here, c can be seen as the size of a single coordination message that each agent needs to send in order to achieve a fully coordinated response. More specifically, the cost c is an expected loss of bandwidth on every channel at each time step per agent, such that $c = 0.01$ in a system of 50 agents implies that one fact less can be posted on each channel every other time step.

[6]The reward is the average achieved per time step during steps 2000 – 2050, to give our learning algorithm time to converge in settings with hundreds of agents. For statistical significance (at the $t < 0.01$ level), we sample each point 500 times.

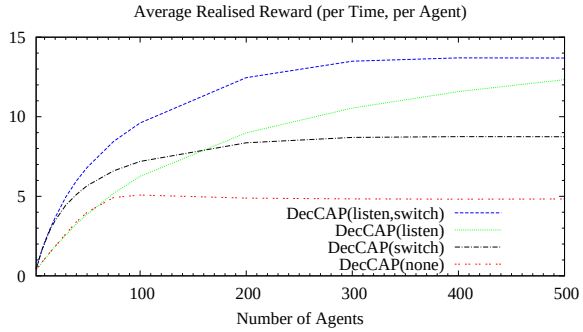


Figure 4: Performance of *DecCAP* variants.

The results for representative costs 0.05, 0.02, 0.01 and 0 (the latter being equivalent to *OPTIMAL/CMLS*) are shown in Figure 3. These demonstrate that approaches using explicit communication perform well in smaller settings, but as soon as the number of agents grows, the communication costs for coordination become non-negligible, and they begin to be outperformed by *DecCAP*.

5.3 Strategy Components

To highlight the benefits of the various components of *DecCAP*, we here briefly evaluate a number of variants of *DecCAP*. As described in Section 4.3, *DecCAP* sometimes forces agents to *listen* to their own channels, i.e., stay silent when this appears more beneficial, while other times agents actively *switch* to channels used by other teams to post particularly valuable facts. We here examine the benefits of these behaviours, using *DecCAP(listen,switch)* to denote a strategy that implements both behaviours, while *DecCAP(listen)* denotes one that implements only the listening, and so on.

The results are shown in Figure 4. This clearly highlights that both the listening and the switching behaviours are key to achieving a high overall performance. Interestingly, the benefit of switching is more pronounced in settings with fewer agents, while listening becomes more important in settings with more agents. Intuitively, this is because fewer facts are discovered in the smaller settings, and so agents benefit from disseminating information to other teams, to fully utilise the available bandwidth. On the other hand, when there are many agents and channels are typically congested, agents gain more from simply listening to the high-value facts posted to their own channels.

5.4 Convergence

Since our strategy relies on learning channel allocations and fact distributions, it takes some time to converge to a good solution. To evaluate how long this takes in practice, Figure 5 shows the performance of *DecCAP* over time, for a small problem with 30 agents

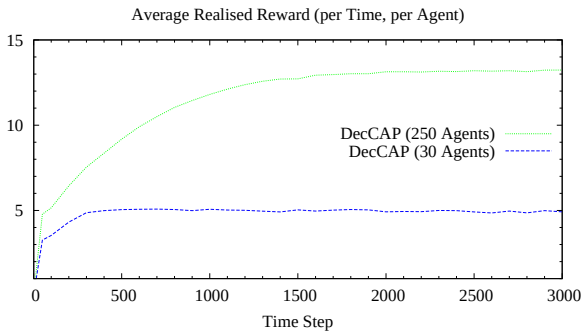


Figure 5: Convergence of *DecCAP* learning.

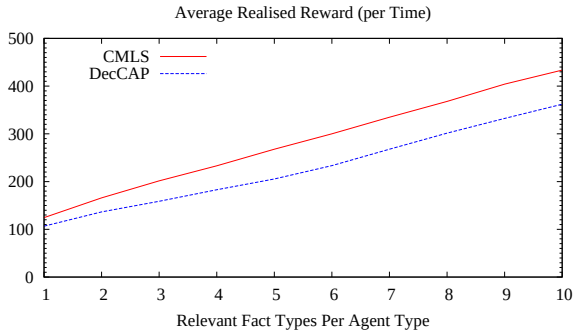


Figure 6: Performance as heterogeneity increases.

and a larger problem with 250 agents. This shows that the strategy converges in a reasonable amount of time. While the maximum in the larger setting is reached after around 2000 time steps, it already achieves 50% of the maximum after 200–300 time steps. In the smaller setting, the maximum is reached more quickly, after around 300 time steps.

5.5 Heterogeneity and Team Synergies

Finally, we consider a more heterogeneous setting where some facts are of interest to multiple types of agents (these synergies also exist in some RoboCupRescue strategies, e.g., when an ambulance uses information about fires to identify particularly critical civilians). We also increase the heterogeneity of the setting under consideration. Thus, we now assume there are five agent types, and each is interested in n out of 10 different types of facts, where we vary n from 1 to 10. As n increases, so do the synergies between teams (more are likely to be interested in the same facts). We also randomly vary the bandwidth of channels, ranging from 0 to 5, we consider 30 agents, and randomly vary the size of agent teams. We choose these parameters to test a more heterogeneous environment where the best performance is not necessarily achieved by allocating each team to a single channel.

The results are given in Figure 6. First, this shows an overall increase in rewards, as each generated fact is increasingly likely to benefit several agent types. Here, *DecCAP* achieves a performance that is within 80–85% of *CMLS*. There is a small drop in reward (relative to *CMLS*) around $n = 5$. This is because *CMLS* can benefit here from re-assigning agents instantaneously between time steps, depending on the overlap of currently known facts. Clearly, such a strategy is not feasible without a central coordinator with full information about all agents (as assumed for *CMLS*). Despite this, *DecCAP* achieves 80% of the centralised near-optimal *CMLS*, indicating that it chooses a suitable channel allocation that performs well in the long run. In more detail, examining the decisions made by *DecCAP* shows that it initially (for $n = 1$) separates different agent types into different channels, but, as fact synergies increase, they increasingly converge to shared channels. This confirms our algorithm flexibly adapts to the problem parameters and communication constraints.

6. CONCLUSIONS

In this paper, we presented an algorithm for channel allocation and information sharing in cooperative agent teams with a highly constrained communication medium. This setting requires the agents not only to reason about who to communicate with and about what, but also how to allocate a restricted communication resource. We present a tractable and fully decentralised learning approach which uses reinforcement learning ideas to learn a channel allocation and then principled decision-theoretic approaches to evaluate the utility of sending pieces of information to others, or even to break from the previously adopted channel allocation, which cannot be done using existing techniques, e.g., in cognitive radio. We compared our approach to benchmarks and showed that it converges quickly to a solution that typically achieves 85% of a centralised optimal strategy.

With this established, we intend to explore the relationship with existing congestion game models including investigating if a finite improvement policy exists, which would allow simple local techniques to converge to Nash equilibria here. This is important since it would allow us to bound our solution quality (which is only empirically demonstrated at present).

Acknowledgement This work was funded by the ALADDIN and ORCHID projects (`www.aladdinproject.org` and `www.orchid.ac.uk`).

7. REFERENCES

[1] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty. Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks*, 50(13):2127–2159, 2006.

[2] A. Alaya-Feki, E. Moulines, and A. LeCornec. Dynamic spectrum access with non-stationary multi-armed bandit. In *SPAWC 2008*, pages 416–420, 2008.

[3] S. Basagni, M. Conti, S. Giordano, and I. Stojmenovic. *Mobile Ad Hoc Networking*. Wiley-Blackwell, 2004.

[4] R. Becker, A. Carlin, V. Lesser, and S. Zilberstein. Analyzing Myopic Approaches for Multi-Agent Communication. *Computational Intelligence*, 25(1):31–50, February 2009.

[5] D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of markov decision processes. In *UAI 2000*, pages 32–37, Stanford, USA, 2000.

[6] G. W. Brown. Iterative solution of games by fictitious play. In T. C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 374–376. New York, 1951.

[7] A. Farinelli, A. Rogers, A. Petcu, and N. R. Jennings. Decentralised coordination of low-power embedded devices using the max-sum algorithm. In *Proc. AAMAS-08*, pages 639–646, May 2008.

[8] K. Hiroaki. Robocup rescue: A grand challenge for multi-agent systems. In *Proc. ICMAS 2000*, pages 5–12, Boston, MA, USA, 2000.

[9] S. Koenig, P. Keskinocak, and C. Tovey. Progress on agent coordination with cooperative auctions. In *Proc. AAAI-10*, pages 1713–1717, 2010.

[10] M. Liu, S. H. A. Ahmad, and Y. Wu. Congestion games with resource reuse and applications in spectrum sharing. In *Proc. GameNets'09*, pages 171–179, 2009.

[11] U. Mir, L. Merghem-Boulahia, and D. Gaiti. A cooperative multiagent based spectrum sharing. *Proc. AICT 2010*, pages 124–130, 2010.

[12] A. Motamedi and A. Baha. Optimal channel selection for spectrum-agile low-power wireless packet switched networks in unlicensed band. *EURASIP Journal on Wireless Communications and Networking*, 2008.

[13] N. Pavlidou, A.J. Han Vinck, J. Yazdani, and B. Honary. Power line communications: state of the art and future trends. *IEEE Commun. Mag.*, 41(4):34–40, 2003.

[14] L. Peshkin, K. Kim, N. Meuleau, and L. Kaelbling. Learning to cooperate via policy search. In *Proc. UAI 2000*, pages 307–314, San Francisco, USA, 2000.

[15] S. A. Williamson, E. H. Gerding, and N. R. Jennings. Reward shaping for valuing communications during multi-agent coordination. In *Proc. AAMAS–09*, pages 641–648, Budapest, Hungary, 2009.

A New Approach to Betweenness Centrality Based on the Shapley Value*

Piotr L. Szczepański
Institute of Informatics, Warsaw University of Technology
Pl. Politechniki 1
00-661 Warsaw, Poland

Tomasz Michalak
Institute of Informatics, University of Warsaw
02-097 Warsaw
ul. Banacha 2, Poland

Talal Rahwan
Electronics and Computer Science, University of Southampton
SO17 1BJ
University Road, UK

ABSTRACT

In many real-life networks, such as urban structures, protein interactions and social networks, one of the key issues is to measure the centrality of nodes, i.e. to determine which nodes and edges are more central to the functioning of the entire network than others. In this paper we focus on *betweenness centrality* — a metric based on which the centrality of a node is related to the number of shortest paths that pass through that node. This metric has been shown to be well suited for many, often complex, networks. In its standard form, the betweenness centrality, just like other centrality metrics, evaluates nodes based on their individual contributions to the functioning of the network. For instance, the importance of an intersection in a road network can be computed as the difference between the full capacity of this network and its capacity when the intersection is completely shut down. However, as recently argued in the literature, such an approach is inadequate for many real-life applications, as, for example, multiple nodes can fail simultaneously. Thus, what would be desirable is to refine the existing centrality metrics such that they take into account not only the functioning of nodes as individual entities but also as members of groups of nodes. One recently-proposed way of doing this is based on the *Shapley Value* — a solution concept in cooperative game theory that measures in a fair way the contributions of players to all the coalitions that they could possibly participate in. Although this approach has been used to extend various centrality metrics, such an extension to betweenness centrality is yet to be developed. The main challenge when developing such a refinement is to tackle the computational complexity; the Shapley Value generally requires an exponential number of operations, making its use limited to a small number of player (or nodes in our context). Against this background, our main contribution in this paper is to refine the betweenness centrality metric based on the Shapley Value: we develop an algorithm for computing this new metric, and show that it has the same complexity as the best known algorithm due to Brandes [7] to compute the standard betweenness centrality (i.e., polynomial in the size of the network). Finally, we show that our results can be extended to another important centrality metric called stress centrality.

*The authors would like to thank all four anonymous reviewers for their useful comments that significantly improved the paper.

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

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous ; D.2.8 [**Software Engineering**]: Metrics—*complexity measures, performance measures*

General Terms

Algorithms, Graph Theory, Game Theory

Keywords

Betweenness Centrality, Shapley Value

1. INTRODUCTION

Networks (or graphs) are a very natural representation of a variety of real-life domains such as, among others, urban structures [20], protein interactions [6], social networks [25] and computer communication networks [13, 27]. Networks are also an important modeling paradigm in Multi-Agent System (see, e.g., [2, 11]). In many of these applications, it is often paramount to determine which nodes (or vertices) and edges are more critical (or *central*) to the functioning of the entire network than others. For example, one may need to know the most influential persons in a social network or the most important routes in a road network. To this end, various centrality metrics, such as *degree*, *closeness*, *eigenvalue* or *betweenness*, have been extensively studied in the literature.[1] In this paper we focus on betweenness centrality — a metric based on which the centrality of a node is related to the number of shortest paths that pass through that node. The importance of this centrality metric stems from the fact that it characterizes well many, often complex and extensive, networks. In particular, a variety of naturally evolving real-life networks, such as the internet or social networks, feature a power-law distribution of betweenness centrality (as well as degree centrality) [3, 13]. Intuitively, in these so called scale-free networks, there are relatively few nodes that contribute to a large number of shortest paths. This implies that such networks are hardly affected by random impairments but, at the same time, they can be relatively easily affected by the removal of the most central nodes [4]. For example, it was shown in the epidemiology literature that the immunization of the most central nodes significantly hinders epidemics [19]. As another example, in the internet context, the betweenness centrality can be used to identify nodes in a local network that are able to trace the communications of as many users as possible [21].

[1] An overview of the most important centrality metrics can be found in Koschützki at al. [18].

Due to a variety of important applications, an ongoing line of research tries to develop better centrality metrics that are more suitable for diverse real world situations [14]. Consider, for example, the problem of designing an infrastructure network, such as a communication network or power grid, where the design is required to be resistant to random node failures. In this respect, centrality metrics, which in this case are called vitality measures [18], simply test the performance of a network with and without a given node. For instance, the importance of an intersection in a road network can be computed as the difference between the full capacity of this network and its capacity when the intersection is completely shut down. Naturally, the more adverse the consequences of a node failure are, the higher the centrality of this node becomes. Nevertheless, such an approach to computing centrality is often inadequate for many real-life network applications.[2] In more detail [1]:

- It is often insufficient to merely consider failures of individual nodes. This is because, in many real-world situations, multiple nodes can fail simultaneously. This was, for instance, the case with the Japanese power grids that were destroyed during the 2011 tsunami. Whereas not much capacity is lost when individual nodes fail in such a network, multiple concurrent failures of certain critical nodes can be highly detrimental. Such issues, however, are not considered in the standard centrality metrics;
- A related impediment of standard centrality metrics is the implicit assumption of node failure independence. This assumption does not hold in many real-world situations when nodes break down sequentially in a short period of time [26].

In an attempt to address the above shortcomings, the notion of *group centrality* [12] has been proposed by which centrality is measured not only for individual nodes but also for groups of them. Whereas, in principle, this notion tackles the issue of group failures, its use is rather limited. This is because it only answers the question of how important a certain group of nodes is. However, it is often unknown *ex ante* which particular groups of nodes should be considered, e.g., in a natural disaster scenario, where it is not possible to identify a priori the nodes that will be affected. Even if we evaluate all possible groups, we are still left without a synthetic ranking of individual nodes' importance. Thus, what would be more desirable is to have a metric of importance of an individual node that takes into account many (preferably all) potential groups that this node can belong to. One such metric is the recently-proposed notion of the *game theoretic network centrality* [15, 24] based on the *Shapley Value*. In more detail, the Shapley Value is one of the key solution concepts in game theory. It advocates a fair division of payoff from a coalitional game and is computed as the weighted average of marginal contributions of players to all the coalitions that they could potentially participate in. The basic idea behind the Shapley Value-based centrality is to define a game over the network, where nodes are players and collections of nodes are coalitions. In this game, the value of any coalition reflects the consequences of a simultaneous failure of all the nodes involved in this coalition. The marginal contribution of a node to a coalition is interpreted as the change in severity of such a failure if this additional node failed with all the other nodes already involved in the coalition. The Shapley Value computed for such a game shows the weighted average of all the marginal contributions of each node to all the potential coalitions of nodes. This addresses both problems related to the standard centrality metrics. Note, however, that computing the Shapley Value in the general case requires a number of operations that is exponential in the number of players. This is clearly not desirable, especially given networks with large numbers of nodes. Thus, to be able to use Shapley Value-based metrics in practice, it is crucial that this complexity is significantly reduced.

Against this background, our contributions in the present paper can be summarised as follows:

1. To date, the Shapley Value was used to refine two standard metrics: degree centrality [24] and closeness centrality [1]. Our first contribution in this paper is to develop the refinement of the betweenness centrality metric based on the Shapley Value;
2. We propose a polynomial-time algorithm for computing the new metric. Interestingly, although our algorithm involves the computation of the Shapley Value, we show that it has the same complexity as the best known algorithm due to Brandes [7] to compute the standard betweenness centrality.
3. Finally, we show that the above results can be extended to another centrality metric, namely the stress centrality [23, 18] — a concept closely related to betweenness centrality.

The remainder of the paper is organized as follows. In Section 2, basic notations and definitions are presented. In Section 3, we introduce the new betweenness centrality based on the Shapley Value. The polynomial time algorithm to compute the new metric is described in Section 4. In Section 5, we summarize our results and present the most promising directions for future research. Finally, a proof is provided in the appendix.

2. PRELIMINARIES

In this section we introduce basic concepts from cooperative game theory that are needed in our analysis.

A *graph* (or *network*) G is composed of *vertices* (or *nodes*) and *edges*. Their sets will be denoted $V(G)$ and $E(G)$, respectively, where every edge in $E(G)$ connects two vertices in $V(G)$. An edge connecting vertices $u, v \in V(G)$ will be denoted by (u, v). We will also consider *weighted networks* in which a weight (label) is associated with every edge in $E(G)$. Informally, a *path* is a sequence of connected edges. The *shortest path* between two given vertices u and v is a path that ends with these vertices and minimizes the weights of the involved edges (or minimizes the number of edges in the case of unweighted networks). It will be denoted by $u \overset{p}{\rightsquigarrow} v$ or simply p if there is no risk of confusion.

Next, we define the notion of a coalitional game and the Shapley Value [22]. In particular, by $A = \{a_1, \ldots, a_{|A|}\}$ we will denote the set of players that participate in a coalitional game. A *characteristic function* $\nu : 2^A \to \mathbb{R}$ assigns to every coalition $C \subseteq A$ a numerical value representing its performance. It is assumed that $\nu(\emptyset) = 0$. A *coalitional game in a characteristic function form* is then a tuple (A, ν). It is usually assumed that the grand coalition, i.e. the coalition of all the players in the game, has the highest value and, therefore, is formed. One of the fundamental questions in cooperative game theory is how to divide the payoff from the grand coalition between the players. Whereas, in principle, there may be an infinite number of such divisions, we are interested in those that meet certain desirable criteria. In this respect, in his highly-regarded paper, Shapley proposed to evaluate the role of each player in the game proportionally to a (weighted) average marginal contribution of this player to all possible coalitions. The importance of the Shapley Value stems from the fact that it is the

[2] A failure of a node should be understood as an inability to perform its previous role, e.g. to control the information flow.

unique division scheme that meets the following four fairness criteria:

1. *Efficiency* — the entire payoff of the grand coalitions is distributed among players;
2. *Symmetry* — all players with the same marginal contributions to all coalitions receive exactly the same payoff;
3. *Null player* — players with no marginal contribution to every coalition receive no payoff; and
4. *Additivity* — values for two given games sum up to the value computed for the sum of both games.

To formalize this concept we denote by $\pi \in \Pi(A)$ a permutation of players in A, and by $P_\pi(i)$ the coalition made of all predecessors of player a_i in π. In more detail, denoting by $\pi(j)$ the location of a_j in π, we have $P_\pi(i) = \{a_j \in \pi : \pi(j) < \pi(i)\}$. Shapley [22] defined the value of a_i, denoted $SV_i(A, \nu)$, as the average marginal contribution of a_i to coalition $P_\pi(i)$ over all $\pi \in \Pi$. Formally:

$$SV_i(A,\nu) = \frac{1}{|A|!}\sum_{\pi\in\Pi}[\nu(P_\pi(i)\cup\{a_i\}) - \nu(P_\pi(i))]. \quad (1)$$

The intuition behind the above formula is as follows: suppose that the players arrive at a certain meeting point in a random order. Furthermore, assume that every player a_i who arrives receives the marginal contribution that this arrival brings to those already at the meeting point. The payoff of player a_i from the coalitional game is then the average of these contributions taken over all the possible orders of arrival.

The above formula can be rewritten into an equivalent form as:

$$SV_i(A,\nu) = \sum_{S\subseteq A\setminus\{a_i\}} \frac{|S|!(|A|-|S|-1)!}{|A|!}[\nu(S\cup\{a_i\}) - \nu(S)]. \quad (2)$$

In the network context, we will denote a coalitional game defined on network G by $(V(G), \nu)$, where a set of vertices is a set of players, $A = V(G)$, and $\nu : 2^{V(G)} \to \mathbb{R}$ is the characteristic function, where $\nu(\emptyset) = 0$. Sometimes, we will use the phrase "value of coalition S" when referring to $\nu(S)$.

3. SHAPLEY VALUE-BASED BETWEENNESS CENTRALITY

In this section we propose the Shapley Value-based betweenness centrality. We start with the definition of the standard betweenness centrality [14]:

Definition 1. The betweenness centrality of a vertex v is defined as a function $c : V \to \mathbb{R} : c_b(v) = \sum_{s\neq v\neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}$, [3] where σ_{st} is the number of shortest paths from s to t (if $s = t$ then $\sigma_{st} = 1$), and $\sigma_{st}(v)$ is the number of shortest paths from s to t passing through vertex v (if $v \in \{s, t\}$ then $\sigma_{st}(v) = 0$).

Intuitively, the betweenness centrality metric represents the load placed on a given vertex in a network. One of the key applications of this metric is to measure the ability of different nodes to control the information flow within a network. However, as we will now show, there are cases where the standard betweenness centrality metric does not produce accurate measurements in such applications. Consider, for example, the network in Figure 1. By computing the centrality of each node in this network using the standard

[3]To deal with unconnected graphs it is assumed that $\frac{0}{0} = 0$.

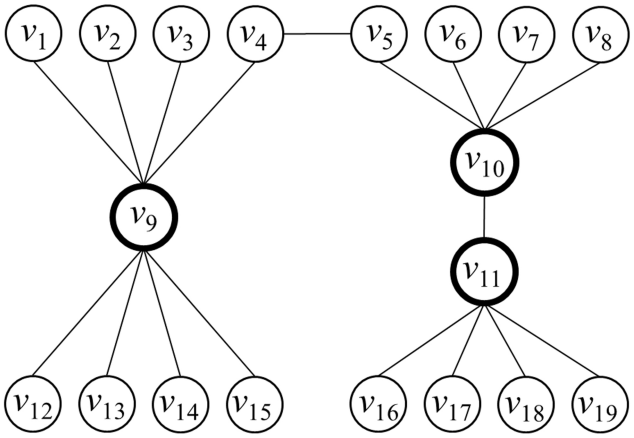


Figure 1: Sample network.

betweenness metric, we find that both v_9 and v_{10} are ranked equally (i.e., $c_b(v_9) = c_b(v_{10}) = 98$)! This is clearly not accurate since the failure of v_9 has more adverse consequences on the ability to pass information through the network than the failure of v_{10}. For example, if v_{10} fails, then the bottom right nodes (i.e., v_{16}, v_{17}, v_{18}, and v_{19}) can still communicate with one another. On the other hand, if v_9 fails, then the bottom left nodes (v_{12}, v_{13}, v_{14}, and v_{15}) can no longer communicate with each other.

In an attempt to deal with this issue, Everett and Borgatti [12] proposed the notion of *group betweenness centrality* of the form:

$$c_{gb}(S) = \sum_{\substack{s\notin S\\ t\notin S}} \frac{\sigma_{st}(S)}{\sigma_{st}},$$

where $S \subseteq V(G)$ is a subset of vertices under consideration, and $\sigma_{st}(S)$ is the number of the shortest paths from s to t passing through some vertex in S (if $s \in S$ or $t \in S$ then $\sigma_{st}(S) = 0$). This centrality metric, however, only evaluates a given subset of vertices S, and this implies that the evaluation of individual nodes remains unchanged. For instance, given the network in Figure 1, group betweenness centrality gives exactly the same ranking of v_9 and v_{10} as the standard betweenness centrality, i.e., $c_{gb}(v_9) = c_{gb}(v_{10}) = 98$ (the only difference is that the former centrality evaluates $\{v_9\}$ and $\{v_{10}\}$, while the latter one evaluates v_9 and v_{10}, respectively; this change does not affect the evaluations of those two nodes). Therefore, even if we evaluate all possible $2^{|V(G)|}$ subsets using group centrality, we are still left without one synthetic ranking of individual vertices' importance.

To address this problem, we now introduce the Shapley Value-based betweenness centrality:

Definition 2. Given a network G, the Shapley Value-based betweenness centrality of a vertex $v \in V(G)$ is defined as a function $c_{Sh} : V \to \mathbb{R} : c_{Sh}(v) = SV_v(V(G), \nu)$, where ν is the characteristic function defined as $\nu : 2^{V(G)} \to \mathbb{R} : \nu(S) = \sum_{\substack{s\notin S\\ t\notin S}} \frac{\sigma_{st}(S)}{\sigma_{st}}$ with $S \subseteq V(G)$.

As mentioned in Section 2, the Shapley Value divides the payoff of the grand coalition among players by evaluating their marginal contributions to any coalition they may possibly belong to. Simply, the higher these marginal contributions are, the higher the Shapley value of a player is. Or, to rephrase it in the context of this paper, the more a vertex contributes to the performance of any possible group of vertices (that this vertex belongs to), the higher

its betweenness centrality should be. Thus, unlike the group betweenness centrality, our Shapley Value-based centrality provides synthetic ranking of individual vertices' importance. Coming back to the example in Figure 1, we find that: $c_{Sh}(v_9) = 18.2$, while $c_{Sh}(v_{10}) = 16.0833$. In other words, our metric is able to reflect the difference in centrality between v_9 and v_{10} because the evaluation of each node is done from a global perspective of all subsets in the network. This approach, among other advantages, grasps a nuance that $\{a_{10}, a_{11}\}$ play the the same role as $\{v_9\}$, and this because $c_{gb}(\{v_9\}) = c_{gb}(\{v_{10}, v_{11}\})$.

Our notion can be seen as analogous to the Shapley Value-based degree and closeness centralities. Having defined it, in the next section we will propose a polynomial time algorithm to compute it.

4. ALGORITHMS TO COMPUTE THE SHAPLEY VALUE BASED BETWEENNESS CENTRALITY

Although the formula for the Shapley Value in (2) is less computationally involved than in (1), it still requires analyzing a number of coalitions that is exponential in the number of players. Specifically, in our network context one would need to analyse $O(2^{|V(G)|})$ coalitions, i.e., groups, of vertices. To circumvent this major obstacle, we propose in this section two polynomial algorithms for computing the Shapley Value-based betweenness centrality: one for weighted graphs, and the other for unweighted graphs. Interestingly, we show that the first algorithm has the same complexity as the best known algorithm to compute the betweenness centrality in the standard form (due to Brandes [7]). Furthermore our both algorithms can be easily adapted to work on directed graphs.

4.1 A Look at Marginal Contributions

Given a graph G and some vertex $v \in V(G)$, we would like to compute the expected marginal contribution of this vertex to the set of vertices $P_\pi(v)$ occurring before v in a random permutation π of all vertices of the graph. We split our analysis into two cases: one of positive and one of negative marginal contributions, respectively.

Firstly, we consider positive contributions. In what follows, let us focus on some particular shortest path p which contains vertex v. Recall that we denote by σ_{st} the number of shortest paths between vertices s and t, and by $\sigma_{st}(v)$ the number of shortest paths between vertices s and t where every path passes through vertex v and $v \neq t \neq s$. Every path in $\sigma_{st}(v)$ has a positive contribution to the coalition $P_\pi(v)$ through v if and only if it is not yet controlled by any vertex from set $P_\pi(v)$. In this case, the positive contribution equals $\frac{1}{\sigma_{st}}$. The necessary and sufficient condition for this to happen can be expressed by $\Psi(p) \cap P_\pi(v) = \emptyset$, where $\Psi(p)$ is the set of all vertices lying on the path p including endpoints. That is, vertices s and t, as well as the rest of the vertices from path p, should not belong to $P_\pi(v)$.

Now, let us introduce a Bernoulli random variable $B^+_{v,p}$ which indicates whether vertex v makes a positive contribution through path p to set $P_\pi(v)$. Thus, we have:

$$\mathbb{E}[\frac{1}{\sigma_{st}} B^+_{v,p}] = \frac{1}{\sigma_{st}} P[\Psi(p) \cap P_\pi(v) = \emptyset],$$

where $P[\cdot]$ denotes probability, and $\mathbb{E}[\cdot]$ denotes expected value. In other words, we need to know the probability of having v precede all other vertices from $\Psi(p) \setminus \{v\}$ in a random permutation of all vertices in the graph. Combinatorial arguments (see the Appendix) show that this happens with probability $\frac{1}{|\Psi(p)|}$. Thus:

$$\mathbb{E}[\frac{1}{\sigma_{st}} B^+_{v,p}] = \frac{1}{\sigma_{st}|\Psi(p)|}. \quad (3)$$

Secondly we examine a potential negative contribution of vertex v to set $P_\pi(v)$. Such a contribution happens when path p ends with v. Specifically, if coalition $P_\pi(v)$ already controls path p, along with vertex v, then not only is there no value added from v becoming a member of this coalition, but there is a negative effect of this move. In particular, the group betweenness centrality assumes that a set of vertices S controls only those paths with both ends not belonging to S. Therefore, when v becomes a member of coalition $P_\pi(v)$, its negative contribution through path p is $-\frac{1}{\sigma_{sv}}$, where, following the previous convention, we denote the number of paths that start with some s end with v by σ_{sv}.

Now, we will analyse a probability of such a negative contribution to happen by considering a complementary event in which path p makes neutral contribution to set $P_\pi(v)$. This happens if and only if either vertex s belongs to set $P_\pi(v)$, or this path is not controlled by any of the vertices in $P_\pi(v)$. Formally: $s \in P_\pi(v) \vee (\Psi(p) \cap P_\pi(v)) = \emptyset$. Now, by introducing a Bernoulli random variable $B^-_{v,p}$ which indicates whether that vertex v makes a negative contribution through path p to set $P_\pi(v)$, we get the following expression:

$$\mathbb{E}[-\frac{1}{\sigma_{sv}} B^-_{v,p}] = -\frac{1}{\sigma_{sv}}(1 - P[s \in P_\pi(v) \vee (\Psi(p) \cap P_\pi(v)) = \emptyset]).$$

Again, one can show with combinatorial arguments that this probability is $P[\Psi(p) \cap P_\pi(v) = \emptyset] = \frac{1}{|\Psi(p)|}$ and due to symmetry that $P[s \in P_\pi(v)] = \frac{1}{2}$. Finally, from the disjointness of these two events, we

$$\mathbb{E}[-\frac{1}{\sigma_{sv}} B^-_{v,p}] = \frac{2 - |\Psi(p)|}{2\sigma_{sv}|\Psi(p)|}. \quad (4)$$

Before proceeding, we define ∂_{st} to be the set of all shortest paths from s to t, and, analogously, $\partial_{st}(v)$ to be the set of shortest paths from s to t passing through vertex v.[4] Now, using the expected value of Bernoulli random variables (3) and (4) we are able to compute the Shapley Value of vertex v, which is the expected marginal contribution of v to $P_\pi(v)$, as:

$$\begin{aligned} SV_v(V(G), \nu) =& \sum_{s \neq v \neq t} \sum_{p \in \partial_{st}(v)} \mathbb{E}[\frac{1}{\sigma_{st}} B^+_{v,p}] + \sum_{s \neq v} \sum_{p \in \partial_{sv}} \mathbb{E}[-\frac{1}{\sigma_{st}} B^-_{v,p}] \\ =& \sum_{s \neq v \neq t} \sum_{p \in \partial_{st}(v)} \frac{1}{\sigma_{st}|\Psi(p)|} + \sum_{s \neq v} \sum_{p \in \partial_{sv}} \frac{2 - |\Psi(p)|}{2\sigma_{sv}|\Psi(p)|}. \end{aligned} \quad (5)$$

The above equation provides insight into the Shapley Value-based betweenness centrality: it is not simply the classical betweenness centrality scaled by the number of vertices that belong to each path. This is because, the second part of the sum resembles the closeness centrality, but with distances measured as the number of vertices on the shortest paths. So, if the vertex lays in the middle of many shortest paths, then its value will be higher.

4.2 The Case of Unweighted Graphs

In this subsection we will construct an efficient algorithm for computing the Shapley Value-based betweenness centrality for unweighted graphs. Specifically, in such graphs, the number of vertices in the shortest path between s and t is simply equal to the distance between s and t, denoted as $d(s,t)$.[5] In other words, we

[4]Note that $\sigma_{st} = |\partial_{st}|$ and $\sigma_{st}(v) = |\partial_{st}(v)|$.

[5]For notational convenience we assume that distance between two vertices is the number of vertices on the shortest path between them (not the number of edges), e.g. $d(s,s) = 1$.

have: $|\Psi(p)| = d(s,t)$. Based on this, it is possible to simplify (5) as follows:

$$SV_v(V(G),\nu) = \sum_{s\neq v\neq t}\sum_{p\in\partial_{st}(v)}\frac{1}{\sigma_{st}d(s,t)} + \sum_{s\neq v}\sum_{p\in\partial_{sv}}\frac{2-d(s,v)}{2\sigma_{sv}d(s,v)}$$
$$= \sum_{s\neq v}\left(\sum_{t\neq v}\frac{\sigma_{st}(v)}{\sigma_{st}d(s,t)} + \frac{2-d(s,v)}{2d(s,v)}\right). \quad (6)$$

The above equation provides some interesting insights: by transforming the second element of the inner sum $\frac{2-d(s,v)}{2d(s,v)} = \frac{1}{d(s,v)} + \frac{1}{2}$ we find that, in unweighted graphs, the Shapley Value using group betweenness centrality as a characteristic function is in fact the sum of *the distanced scaled betweenness centrality* (introduced by Borgatti and Everett in [5]) and the *closeness* centrality, shifted by half.

Now, we adopt the framework presented in [7] so as to accommodate equation (6). We denote by $\delta_{s,t}(v) = \frac{\sigma_{st}(v)}{d(s,t)\sigma_{st}}$ a pair-dependency, which is the positive contribution that vertices s and t make to the assessment of vertex v in equation (6). Analogously, we denote by $\delta_{s,\cdot}(v) = \sum_{t\in V}\delta_{s,t}(v)$ one-side dependency, which is the positive contribution that vertex s makes to the evaluation of vertex v in the equation (6).

A naive way to compute the betweenness centrality is to first compute the number of shortest paths between all pairs, and then sum all pair-dependencies. This process takes $O(|V|^3)$ time. Brandes [7] proposed an algorithm to improve this complexity by using some recursive relation. This algorithm runs in $O(|V|\cdot|E|)$ time, and requires $O(|V|+|E|)$ space. We will now show that, although our new centrality is based on the Shapley Value, it can be computed with the same complexity as Brandes's algorithm.

Building upon Brandes [7], and its modification for distanced scaled betweenness centrality presented in [8] we have:

$$\delta_{s,\cdot}(v) = \sum_{\substack{w:\ (v,w)\in E \\ d(s,w)=d(s,v)+1}}\frac{\sigma_{sv}}{\sigma_{sw}}\left(\frac{1}{d(s,w)} + \delta_{s,\cdot}(w)\right). \quad (7)$$

Now, we are able to compute our Shapley Value-based betweenness centrality for a vertex v by iterating over all other vertices and summing their contributions. Using (6) and (7) we get:

$$SV_v(V(G),\nu) = \sum_{s\neq v}\left(\delta_{s,\cdot}(v) + \frac{2-d(s,v)}{2d(s,v)}\right). \quad (8)$$

Algorithm 1 modifies Brandes's approach and computes the Shapley Value-based betweenness centrality. It runs in $O(|V|\cdot|E|)$ time, and requires $O(|V|+|E|)$ space.

Firstly, in lines 7 - 15, the algorithm calculates both the distance and the number of shortest paths from a source s to each vertex. While doing that, for each vertex v, all directly preceding vertices occurring on shortest paths from s to v are stored in memory. This process uses Breadth-First Search [10] which takes $O(|V|)$ time and $O(|V|+|E|)$ space. In the second step (lines 20 and 22), the algorithm uses formula (8) to calculate the contribution of the source s to the value of our betweenness centrality for each vertex that is reachable from the source. This step also takes $O(|V|)$ time and $O(|V|+|E|)$ space.

As visible in formula (8), in an undirected graph, each path is considered twice. Thus in line 20 which is inside the loop we multiply the influence of the vertex s by two. At the end of the algorithm, in line 23, we halve the accumulated result. Finally, we note that it is very easy to adopt Algorithm 1 to directed graphs. To this end, we remove the loop from line 23 and halve the contribution of the vertex s from line 22, which now should look as follows:

22 : $c_{Sh}(w) \leftarrow c_{Sh}(w) + \delta_{s,\cdot}(w) + \frac{2-d(s,w)}{2d(s,w)}$;

Algorithm 1: Computing Shapley Value-based betweenness centrality for unweighted graphs

Input: Graph $G = (V, E)$
Data: queue $\mathcal{Q}$, stack $\mathcal{S}$ for each $v \in V$ and some source s:
$d(s,v)$: distance from v to the source s
$Pred_s(v)$: list of predecessors of v on the shortest paths from source s
σ_{sv} : the number of shortest paths from s to v
$\delta_{s,\cdot}(v)$: the one-side dependency of s on v
Output: $c_{Sh}(v)$ Shapley Value-based betweenness centrality for each vertex $v \in V$

1 **foreach** $v \in V$ **do**
2 $\quad c_{Sh}(v) \leftarrow 0$;
3 **foreach** $s \in V$ **do**
4 $\quad$ **foreach** $v \in V$ **do**
5 $\quad\quad Pred_s(v) \leftarrow empty\ list;\ d(s,v) \leftarrow \infty;\ \sigma_{sv} \leftarrow 0$;
6 $\quad d(s,s) \leftarrow 1;\ \sigma_{ss} \leftarrow 1;\ \ enqueue\ s \rightarrow \mathcal{Q}$;
7 $\quad$ **while** $\mathcal{Q}$ *is not empty* **do**
8 $\quad\quad dequeue\ v \leftarrow \mathcal{Q};\ push\ v \rightarrow \mathcal{S}$;
9 $\quad\quad$ **foreach** w *such that* $(v,w) \in E$ **do**
10 $\quad\quad\quad$ **if** $d(s,w) = \infty$ **then**
11 $\quad\quad\quad\quad d(s,w) \leftarrow d(s,v) + 1$
12 $\quad\quad\quad\quad enqueue\ w \rightarrow \mathcal{Q}$
13 $\quad\quad\quad$ **if** $d(s,w) = d(s,v) + 1$ **then**
14 $\quad\quad\quad\quad \sigma_{sw} \leftarrow \sigma_{sw} + \sigma_{sv}$;
15 $\quad\quad\quad\quad append\ v \rightarrow Pred_s(w)$;
16 $\quad$ **foreach** $v \in V$ **do** $\delta_{s,\cdot}(v) \leftarrow 0$;
17 $\quad$ **while** $\mathcal{S}$ *is not empty* **do**
18 $\quad\quad pop\ w \leftarrow \mathcal{S}$;
19 $\quad\quad$ **foreach** $v \in Pred_s(w)$ **do**
20 $\quad\quad\quad \delta_{s,\cdot}(v) \leftarrow \delta_{s,\cdot}(v) + \frac{\sigma_{sv}}{\sigma_{sw}}(\frac{1}{d(s,w)} + \delta_{s,\cdot}(w))$;
21 $\quad\quad$ **if** $w \neq s$ **then**
22 $\quad\quad\quad c_{Sh}(w) \leftarrow c_{Sh}(w) + \delta_{s,\cdot}(w) + \frac{2-d(s,w)}{d(s,w)}$;
23 **foreach** $v \in V$ **do**
24 $\quad c_{Sh}(v) = \frac{c_{Sh}(v)}{2}$;

4.3 The Case of Weighted Graphs

While the focus of the previous subsection was on unweighted graphs, in this subsection we show how to compute the Shapley Value-based betweenness centrality for weighted graphs. In particular, we consider one of the most popular semantics of weighted graphs, where the weight $\lambda(v,u)$ of the edge between v and u is interpreted as the distance between v and u . Thus, it is very likely that for some shortest path $s \overset{p}{\rightsquigarrow} t$ it holds that $|\Psi(p)| \neq d(s,t)$.

We will denote by Υ_{st} the sum of the reciprocals of the number of vertices belonging to all particular shortest paths between vertices s and t. Formally:

$$\Upsilon_{st} = \sum_{p\in\partial_{st}}\frac{1}{|\Psi(p)|}. \quad (9)$$

Furthermore, following our convention, we also define $\Upsilon_{st}(v) =$

$\sum_{p\in\partial_{st}(v)} \frac{1}{|\Psi(p)|}$. Now, using (9) it is possible to simplify (5) as follows:

$$SV_v(V(G),\nu) = \sum_{s\neq v}\left(\sum_{t\neq v}\frac{\Upsilon_{st}(v)}{\sigma_{st}} + \frac{\Upsilon_{sv}}{\sigma_{sv}} - \frac{1}{2}\right). \quad (10)$$

In order to compute this value efficiently, we need to overcome two main algorithmic challenges. The first is how to efficiently compute Υ_{st} for each s and t. The second challenge is how to recursively compute the term $\sum_{t\neq v}\frac{\Upsilon_{st}(v)}{\sigma_{st}}$, which is the one-side dependency in weighted graphs (denoted as $\delta_{s,\cdot}(v)$). That is,

$$\delta_{s,\cdot}(v) = \sum_{t\in V}\frac{\Upsilon_{st}(v)}{\sigma_{st}}. \quad (11)$$

In the above equation, counting all shortest paths between source s and each vertex t, as well as the number of vertices in each such path, is not challenging: it can be done using $O(V^2)$ space. However, it is not clear whether there exists any recursive relation that computes (11), i.e. a similar relation to that used in (7).

In order to compute (11) recursively, we will define an array T_{st} which stores the number of shortest paths between vertices s and t. as well as the number of vertices in each such path. More specifically, $T_{st}[i] : i \in \{1,\ldots,|V|\}$ is the number of shortest paths between s and t that contain exactly i vertices. The array T_{st} uniquely determines the polynomial W_{st} with terms $T_{st}[i]x^i$. We define seven operation on such arrays:

Shifting $T_{st}^{\rightarrow}$ and $T_{st}^{\leftarrow}$ increase or decrease the indices of all values of the array by one, respectively. This takes $O(|V|)$ time.

Evaluating $\|T_{st}\|$ returns $\sum_{i=1}^{|V|}\frac{T_{st}[i]}{i}$. Time complexity is $O(|V|)$.

Adding $T_{sv}\oplus T_{su}$ is an operation of adding two polynomials W_{sv} and W_{su}. It takes $O(|V|)$ time. We will denote by $\bigoplus$ the sum of a series of polynomials.

Multiplying $T_{sv}\otimes T_{vt}$ is an operation of multiplying two polynomials W_{sv} and W_{vt}. This takes $O(|V|\log|V|)$ time using the polynomial multiplying algorithm from [10].

Dividing $T_{sv}\oslash T_{vt}$ is an operation of dividing polynomials W_{sv} and W_{vt}. This takes $O(|V|\log|V|)$ time.

Dividing by real $T_{sv}\div k$ means dividing every value in the array by the real value k. This operation takes $O(|V|)$ time.

Resetting $T_{sv}\leftarrow 0$ is an operation that assigns 0 to each cell in T_{sv}.

Observe that $\|T_{st}\| = \Upsilon_{st}$. Therefore, to overcome the first algorithmic challenge, it is sufficient to compute $\|T_{st}\|$. We will use the following relation:

$$T_{sv} = \bigoplus_{\substack{u:\ d(s,u)+\\ \lambda(u,v)=d(s,v)}} T_{su}^{\rightarrow}. \quad (12)$$

Using Dijkstra's algorithm [10], as well as equation (12), we can compute T_{st} for every t and some source s. If vertex u precedes vertex v on some shortest path from source s, all shortest paths stored in T_{su} extended by vertex v are part of the set of shortest paths stored in T_{sv}. This procedure takes $O(|V|^2|E| + |V|^2\log|V|)$ time.

To solve the second algorithmic challenge, it is necessary to notice the following relationship:

$$T_{st}(v) = (T_{sv}\otimes T_{vt})_{st}^{\leftarrow} = T_{sv}\otimes T_{vt}^{\leftarrow}. \quad (13)$$

Following our convention, $T_{st}(v)$ is an array that stores information about the paths between s and t that pass through the vertex v. Every path stored in the array T_{sv} can be extended by every path stored in the array T_{vt}. This operation, which is in fact the multiplication of two polynomials W_{sv} and W_{vt}, gives us information about all shortest paths from s to t passing through v. The vertex v is counted twice, so by shifting left the result of multiplication we shorten all paths by one.

Now, we are able to infer the recursive relation. Changing type of one-side dependency (11) to the type of the proposed array $\delta^*_{s,\cdot}(v) = \bigoplus_{t\in V}\frac{T_{st}(v)}{\sigma_{st}}$, using (13), and using the property of a polynomial operation, we obtain the following relation [6]:

$$\delta^*_{s,\cdot}(v) = \bigoplus_{\substack{w:\ d(s,v)+\\ \lambda(v,w)=d(s,w)}}\left(\frac{T_{sv}^{\rightarrow}}{\sigma_{sw}}\oplus T_{sv}\otimes(\delta^*_{s,\cdot}(w)\oslash T_{sw}^{\leftarrow})\right). \quad (14)$$

Equations (10) and (11), and definition of $\delta^*_{s,\cdot}(v)$ give us the ultimate formula:

$$SV_v(V(G),\nu) = \sum_{s\neq v}\left(\left\|\delta^*_{s,\cdot}(v)\right\| + \frac{\|T_{sv}\|}{\sigma_{sv}} - \frac{1}{2}\right). \quad (15)$$

We use the above result to construct Algorithm 2 that computes the Shapley Value-based betweenness centrality for weighted graphs in $O(|E|\cdot|V|^2\log|V|)$ time. The algorithm require $O(|V|^2)$ space.

Algorithm 2 shows great similarity to Algorithm 1. The only difference is that we do not operate on numbers of shortest paths between vertices, but on the special array introduced, which is the reason behind the higher complexity. However, analogously to Algorithm 1, we are able to easily adapt this algorithm to work on directed graphs. It is necessary to remove the loop from line 27 and halve the contribution of vertex s from line 26. Thus, for the case of directed graphs, this lines becomes:

26 : $c_{Sh}(w) \leftarrow c_{Sh}(w) + \left\|\delta^*_{s,\cdot}(w)\right\| + \frac{\|T_{sw}\|}{\sigma_{sw}} - \frac{1}{2}$;

4.4 Shapley Value-based Stress Centrality

We will now show how to adapt Algorithms 1 and 2 so that they efficiently compute the stress centrality based on the Shapley Value. Intuitively, the stress centrality [23, 18] is used to identify the vertices that are exposed to high loads. Formally,

Definition 3. The stress centrality of node v is defined as a function $c : V\to\mathbb{R} : c_s(v) = \sum_{s\neq v\neq t}\sigma_{st}(v)$.

Although the stress centrality has a very similar functional form to the betweenness centrality, both metrics may rank nodes differently. The above definition of centrality can be refined to the Shapley Value-based stress centrality, as follows:

Definition 4. Given network G, the Shapley Value-based stress centrality of vertex $v\in V(G)$ is defined as a function $c_{Sh} : V\to\mathbb{R} : c_{Sh}(v) = SV_v(V(G),\nu)$, where ν is the characteristic function defined as $\nu : 2^{V(G)}\to\mathbb{R} : \nu(S) = \sum_{\substack{s\notin S\\ t\notin S}}\sigma_{st}(S)$ with $S\subseteq V(G)$.

[6] We omit the precise description of a derivation which is analogous to Brandes' derivation of the equation (7).

where group stress centrality is defined in an analogous way to that of the group betweenness centrality in Section 3. The analysis of the expected marginal contribution of some vertex $v \in V(G)$ to the set of vertices $P_\pi(v)$ in the context of group stress centrality c leads to the following equation:

$$SV_v(V(G), c) = \sum_{s \neq v \neq t} \sum_{p \in \partial_{st}(v)} \mathbb{E}[B^+_{v,p}] + \sum_{s \neq v} \sum_{p \in \partial_{sv}} \mathbb{E}[-B^-_{v,p}]$$
$$= \sum_{s \neq v \neq t} \sum_{p \in \partial_{st}(v)} \frac{1}{|\Psi(p)|} + \sum_{s \neq v} \sum_{p \in \partial_{sv}} \frac{2 - |\Psi(p)|}{2|\Psi(p)|}, \quad (16)$$

where we use the same notation as in Section 4.1.

Algorithm 2: Computing Shapley Value-based betweenness centrality for weighted graphs

Input: weighted graph $G = (V, E)$, with weight function $\lambda : E \to \mathbb{R}^+$
Data: priority queue $\mathcal{Q}$ with key $d()$, stack $\mathcal{S}$ for each vertex $v \in V$ and some source s:
$d(s, v)$: the distance from s to v
$Pred_s(v)$: the list of predecessors of v on the shortest paths from source s
σ_{sv} : the number of shortest paths from s to v
$\delta^*_{s,\cdot}(v)$: one-side dependency of s on v with type of the array
T_{sv} : the number of shortest paths from s to v with accuracy to the number of vertices belonging to them stored in array
Output: $c_{Sh}(v)$ Shapley Value-based betweenness centrality

1 **foreach** $v \in V$ **do**
2 $\quad c_{Sh}(v) \leftarrow 0$;
3 **foreach** $s \in V$ **do**
4 $\quad$ **foreach** $v \in V$ **do**
5 $\quad\quad Pred_s(v) \leftarrow empty\ list$; $d(s, v) \leftarrow \infty$; $\sigma_{sv} \leftarrow 0$;
6 $\quad d(s, s) \leftarrow 1$; $\sigma_{ss} \leftarrow 1$; $enqueue\ s \to \mathcal{Q}$;
7 $\quad$ **while** $\mathcal{Q}$ *is not empty* **do**
8 $\quad\quad extract\ v \leftarrow \mathcal{Q}$ with minimal $d(s, v)$;
9 $\quad\quad push\ v \to \mathcal{S}$;
10 $\quad\quad$ **foreach** w *such that* $(v, w) \in E$ **do**
11 $\quad\quad\quad$ **if** $d(s, w) > d(s, v) + \lambda(v, w)$ **then**
12 $\quad\quad\quad\quad d(s, w) \leftarrow d(s, v) + \lambda(v, w)$
13 $\quad\quad\quad\quad insert/update\ w \to \mathcal{Q}$ with $d(s, w)$;
14 $\quad\quad\quad\quad \sigma_{sw} \leftarrow 0$; $T_{sw} \leftarrow 0$;
15 $\quad\quad\quad\quad Pred_s(w) \leftarrow empty\ list$;
16 $\quad\quad\quad$ **if** $d(s, w) = d(s, v) + \lambda(v, w)$ **then**
17 $\quad\quad\quad\quad \sigma_{sw} \leftarrow \sigma_{sw} + \sigma_{sv}$;
18 $\quad\quad\quad\quad append\ v \to Pred_s(w)$;
19 $\quad\quad\quad\quad T_{sw} = T_{sw} \oplus T^{\rightarrow}_{sv}$;
20 $\quad$ **foreach** $v \in V$ **do** $\delta^*_{s,\cdot}(v) \leftarrow 0$;
21 $\quad$ **while** $\mathcal{S}$ *is not empty* **do**
22 $\quad\quad pop\ w \leftarrow \mathcal{S}$;
23 $\quad\quad$ **foreach** $v \in Pred_s(w)$ **do**
24 $\quad\quad\quad \delta^*_{s,\cdot}(v) \leftarrow \delta^*_{s,\cdot}(v) \oplus \frac{T^{\rightarrow}_{sv}}{\sigma_{sw}} \oplus T_{sv} \otimes (\delta^*_{s,\cdot}(w) \oslash T^{\leftarrow}_{sw})$;
25 $\quad\quad$ **if** $w \neq s$ **then**
26 $\quad\quad\quad c_{Sh}(w) \leftarrow c_{Sh}(w) + \left\|\delta^*_{s,\cdot}(w)\right\| + \frac{2\|T_{sw}\|}{\sigma_{sw}} - 1$
27 **foreach** $v \in V$ **do**
28 $\quad c_{Sh}(v) = \frac{c_{Sh}(v)}{2}$;

Using (16), and following similar steps to those that we took during the analysis of the betweenness centrality, we infer analogous recursive equations for computation one-side dependency $\delta_{s,\cdot}(v)$. In case of unweighted graphs, we get $\delta_{s,\cdot}(v) = \sum_{t \in V(G)} \frac{\sigma_{st}(v)}{d(s,t)}$ and obtain:

$$\delta_{s,\cdot}(v) = \sum_{\substack{w:\ (v,w) \in E \\ d(s,w) = d(s,v)+1}} \sigma_{sv} \left(\frac{1}{d(s,w)} + \frac{\delta_{s,\cdot}(w)}{\sigma_{sw}} \right). \quad (17)$$

The modification of Algorithm 1 based on equations (16) and (17) consists of changing lines 20 and 22 so that they become:

20 : $\delta_{s,\cdot}(v) \leftarrow \delta_{s,\cdot}(v) + \sigma_{sv}(\frac{1}{d(s,w)} + \frac{\delta_{s,\cdot}(w)}{\sigma_{sw}})$;
22 : $c_{Sh}(w) \leftarrow c_{Sh}(w) + \delta_{s,\cdot}(w) + \sigma_{sw}(\frac{2-d(s,w)}{d(s,w)})$;

Then, in case of weighted graphs, where $\delta^*_{s,\cdot}(v) = \bigoplus_{t \in V} T_{st}(v)$ we obtain:

$$\delta^*_{s,\cdot}(v) = \bigoplus_{\substack{w:\ d(s,v)+ \\ \lambda(v,w) = d(s,w)}} \left(T^{\rightarrow}_{sv} \oplus T_{sv} \otimes (\delta^*_{s,\cdot}(w) \oslash T^{\leftarrow}_{sw}) \right). \quad (18)$$

Equations (16) and (18) result in changing lines 24 and 26 from Algorithm 2 into the following:

24 : $\delta^*_{s,\cdot}(v) \leftarrow \delta^*_{s,\cdot}(v) \oplus T^{\rightarrow}_{sv} \oplus T_{sv} \otimes (\delta^*_{s,\cdot}(w) \oslash T^{\leftarrow}_{sw})$;

26 : $c_{Sh}(w) \leftarrow c_{Sh}(w) + \left\|\delta^*_{s,\cdot}(w)\right\| + 2\left\|T_{sw}\right\| - \sigma_{sw}$;

This concludes the necessary modifications of Algorithms 1 and 2 in order to compute the Shapley Value-based stress centrality.

5. SUMMARY AND FUTURE WORK

In Table 1, we present a summary of the results obtained in this paper. To date, following the seminal work of Gómez et al. [15], the game theoretic refinements for the degree [24] and closeness [1] centralities were proposed and their computational properties were studied in Aadithya et al. [1]. In the present paper we propose the Shapley Value-based betweenness centrality and develop two polynomial algorithms for computing it. We also show that these results can be easily extended to the related notion of the stress centrality.

Standard centrality	Group centrality	SV-based centrality	Efficient computation
node	[12]	[24]	[1]
closeness	[12]	[1]	[1]
betweenness	[12]	this paper	this paper
stress	this paper	this paper	this paper

Table 1: Summary of the results obtained in this paper.

Regarding future research, our work can be extended to a variety of other centrality metrics. In particular, similarly to the stress centrality, there are other, though less known, versions of the betweenness centrality [8] to which our results stretch straightforwardly. Another interesting, and indeed more challenging, extension is to derive (and efficiently compute!) the Shapley Value-based forms of other metrics such as the graph centrality [17], the reach centrality [16], the edge centrality [18], the flow betweenness centrality [18], and current flow centrality [9].

6. REFERENCES

[1] K.V. Aadithya, B. Ravindran, T.P. Michalak, and N.R. Jennings, *Efficient Computation of the Shapley Value for Centrality in Networks*, WINE'10, 2010.

[2] Satomi Baba, Atsushi Iwasaki, Makoto Yokoo, Marius-Calin Silaghi, Katsutoshi Hirayama, and Toshihiro Matsui, *Cooperative problem solving against adversary: quantified distributed constraint satisfaction problem*, AAMAS, 2010, pp. 781–788.

[3] M. Barthelemy, *Betweenness centrality in large complex networks*, The European Physical Journal B - Condensed Matter **38** (2004), no. 2, 163–168.

[4] B. Bollobas and O. Riordan, *Robustness and vulnerability of scale-free random graphs* , Internet Mathematics **1** (2003), no. 1, 1–35.

[5] S. P. Borgatti and M. G. Everett, *A Graph-theoretic perspective on centrality* , Social Networks **28** (2006), no. 4, 466–484.

[6] P. Bork, L. J. Jensen, von C. Mering, A. K. Ramani, I. Lee, and E. M. Marcott, *Protein interaction networks from yeast to huma*, Curr. Opin. Struct. Biol. **14** (2004), no. 3, 292–299.

[7] U. Brandes, *A faster algorithm for betweenness centrality*, J. of Mathematical Sociology **25** (2001), no. 2, 163–177.

[8] ———, *On variants of shortest-path betweenness centrality and their generic computation* , Social Networks **30** (2008), no. 2, 136–145.

[9] Ulrik Brandes and Daniel Fleischer, *Centrality measures based on current flow*, 2005.

[10] T.H. Cormen, *Introduction to algorithms*, MIT Press, 2001.

[11] Mathijs M. de Weerdt and Yingqian Zhang, *Preventing under-reporting in social task allocation*, Proceedings of the 10th workshop on Agent-Mediated Electronic Commerce (AMEC-X) (Han La Poutre and Onn Shehory, eds.), IFAAMAS, 2008.

[12] M. G. Everett and S. P. Borgatti, *The centrality of groups and classes* , Journal of Mathematical Sociology **23** (1999), no. 3, 181–201.

[13] M. Faloutsos, P. Faloutsos, and Faloutsos C., *On power-law relationships of the internet topology*, SIGCOMM Comput. Comm. Rev. **29** (1999), no. 4, 251–262.

[14] L.C. Freeman, *Centrality in social networks: Conceptual clarification*, Social Networks **1** (1979), no. 3, 215–239.

[15] D. Gómez, E. González-Arangüena, C. Manuel, G. Owen, M. Del Pozo, and J. Tejada, *Centrality and power in social networks: A game theoretic approach*, Mathematical Social Sciences **46** (2003), no. 1, 27–54.

[16] R. J. Gutman, *Reach-Based Routing: A New Approach to Shortest Path Algorithms Optimized for Road Networks*, In Proceedings of the 6th Workshop on Algorithm Engineering and Experiments, SIAM, 2004, pp. 100–111.

[17] P. Hage and F. Harary, *Eccentricity and centrality in networks.* , Social Networks **17** (1995), 57–63.

[18] D. Koschützki, K.A. Lehmann, L. Peeters, S. Richter, D. Tenfelde-Podehl, and O. Zlotowski, *Centrality indices. network analysis*, Lecture Notes in Computer Science, vol. 3418, pp. 16–61, Springer, 2005.

[19] R. Pastor-Satorras and A. Vespignani, *Immunization of complex networks* , Phys. Rev. E **65:036104** (2002).

[20] S. Porta, P. Crucitti, and V. Latora, *The network analysis of urban streets: a primal approach*, Environment and Planning B: Planning and Design **33** (2006), no. 5, 705–725.

[21] R. Puzis, D. Yagil, Y. Elovici, and D. Braha, *Collaborative attack on internet users' anonymity* , Internet Research **19** (2009), no. 1, 60–77.

[22] L. S. Shapley, *A value for n-person games*, In Contributions to the Theory of Games, volume II (H.W. Kuhn and A.W. Tucker, eds.), Princeton University Press, 1953, pp. 307–317.

[23] A. Shimbel, *Structural parameters of communication networks* , Bull. of Math. Biophysics **15** (1953), 501–507.

[24] N.R. Suri and Y. Narahari, *Determining the top-k nodes in social networks using the Shapley Value*, AAMAS '08: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems, 2008, pp. 1509–1512.

[25] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*, England: Cambridge University Press., 1994.

[26] D.J. Watts and S.H. Strogatz, *Collective dynamics of small-world networks*, Nature **393** (1998), no. 6684, 440–442.

[27] S. H. Yook, H. Jeong, and A.-L. Barabasi, *Modeling the internet's large-scale topology*, Proceedings of the National Academy of Science **99** (2002), no. 21, 13382–13386.

APPENDIX: Combinatorial Proof

Theorem 1. Let K be a set of elements such that $|K| = k$. Let L and R be two disjoint subsets of K, such that: $|L| = l, |R| = r$. Now, given some element $x \in K$, where $x \notin L \cup R$, and given a random permutation $\pi \in \Pi(K)$, the probability of having every element in L before x, and every element in R after x, is:

$$P[\forall_{e \in L} \pi(e) < \pi(x) \wedge \forall_{e \in R} \pi(e) > \pi(x)] = \frac{1}{(l+1)\binom{l+r+1}{r}}$$

PROOF. Let us first count the permutations that satisfy the assumption: $\forall_{e \in L} \pi(e) < \pi(x) \wedge \forall_{e \in R} \pi(e) > \pi(x)$. Specifically:

- Let us choose $l + r + 1$ positions in the sequence of all elements from K. There are $\binom{n}{l+r+1}$ such possibilities.
- Now, in the first l chosen positions, place all elements from L. Directly after those, place the element x. Finally, in the last r chosen positions, place all elements from R. The number of such line-ups is $l!r!$.
- The remaining elements can be arrange in $(n - (l + r + 1))!$ different possibilities.

Thus, the number of permutations satisfying our assumption is:

$$\binom{n}{l+r+1} l!r!(n - (l + r + 1))! = \frac{n!}{(l+1)\binom{l+r+1}{r}},$$

□

From Theorem 1 we can obtain the probability of an event in which the vertex v laying on the path p precedes all the other vertices from this path in a random permutation of all vertices in the graph G. Now, by setting $K = V(G)$, $L = \emptyset$ and $R = \Psi(p) \setminus \{v\}$, we obtain the desired probability: $\frac{1}{|\Psi(p)|}$.

Maintaining Team Coherence under the Velocity Obstacle Framework

Andrew Kimmel
University of Nevada, Reno
1664 N. Virginia St., MS171
Reno, NV 89557
akimmel@cse.unr.edu

Andrew Dobson
University of Nevada, Reno
1664 N. Virginia St., MS171
Reno, NV 89557
dobsona@cse.unr.edu

Kostas Bekris
University of Nevada, Reno
1664 N. Virginia St., MS171
Reno, NV 89557
bekris@cse.unr.edu

ABSTRACT

Many multi-agent applications may involve a notion of spatial coherence. For instance, simulations of virtual agents often need to model a coherent group or crowd. Alternatively, robots may prefer to stay within a pre-specified communication range. This paper proposes an extension of a decentralized, reactive collision avoidance framework, which defines obstacles in the velocity space, known as Velocity Obstacles (VOs), for coherent groups of agents. The extension, referred to in this work as a Loss of Communication Obstacle (LOCO), aims to maintain proximity among agents by imposing constraints in the velocity space and restricting the set of feasible controls. If the introduction of LOCOs results in a problem that is too restrictive, then the proximity constraints are relaxed in order to maintain collision avoidance. If agents break their proximity constraints, a method is applied to reconnect them. The approach is fast and integrates nicely with the Velocity Obstacle framework. It yields improved coherence for groups of robots connected through an input constraint graph that are moving with constant velocity. Simulated environments involving a single team moving among static obstacles, as well as multiple teams operating in the same environment, are considered in the experiments and evaluated for collisions, computational cost and proximity constraint maintenance. The experiments show that improved coherence is achieved while maintaining collision avoidance, at a small computational cost and path quality degradation.

Categories and Subject Descriptors

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

General Terms

Algorithms, Design, Experimentation

Keywords

AAMAS proceedings, multi-agent, collision avoidance, team coherence, communication constraints

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

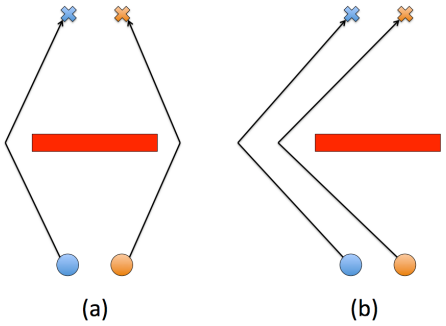


Figure 1: Two agents navigating around a static obstacle: (a) the agents split to reach their goals without collisions, while in (b) the agents move together as a coherent team. The second behavior must be achieved in a decentralized manner.

1. INTRODUCTION

Many practical applications of decentralized collision avoidance may involve a secondary objective, where teams of agents need to maintain a certain level of coherence. Coherence often implies that the agents should remain within a certain distance of one another. In games and simulations, the agents may need to remain within a certain distance because of implied social interactions, or because they need to reach their destination together so as to be more effective in completing an objective at their goal. For instance, a team of agents in a game will be more effective in attacking an enemy if all the units move together against the opponent and do not split into multiple groups. In mobile sensor networks, robots may have to respect radial communication limits.

The Velocity Obstacle (VO) formulation [8, 27, 22] is a framework for reactive collision avoidance. It is fast as it operates directly in the velocity space of each agent. It is also a decentralized approach as each agent reasons independently about its controls, as long as it can compute the position and velocity of its neighbors. Current work in Velocity Obstacles does not directly address the issue of coherence. Consider the situation in Figure 1, where two agents are required to avoid an obstacle and reach their desired destination. An application of the basic VO framework may result in the two agents splitting and passing the obstacle from opposite ends. It would be desirable, however, for the agents to select, in a decentralized manner, a single direction to follow so as to avoid the obstacle and reach their goal while maintaining

coherence. The decentralized nature of the solution will be especially helpful in robotic applications because no communication will be required between robots. It will also provide improved scalability in simulations and games.

This paper proposes a method for maintaining team coherence within the VO framework in a decentralized manner. The desired coherence for a problem can be defined as a graph of dependencies between agents. An edge in the graph implies that the corresponding agents should remain within a predefined distance as they move towards their goal. Given this input graph, an agent constructs an additional obstacle in the velocity space for each neighbor with which it wants to maintain connectivity. This construction is referred to in this work as a Loss of Communication Obstacle (LOCO).

Building on top of the VO framework allows LOCOs to focus on coherence maintenance, rather than obstacle avoidance. In order to construct the LOCO, the assumption is that a neighbor will maintain its current control, as in the original VO framework. Then, a LOCO defines the set of velocities that will lead the two agents to be separated beyond a desired distance within a certain time horizon. A scheme is proposed for integrating information from the multiple VOs defined for collision avoidance and the LOCOs defined for coherence maintenance. If the set of velocities that satisfy both constraints is not empty for a satisfactory time horizon, then the velocity in this set that brings the agent closer to its goal is chosen. A valid velocity implies that, given the neighbor does not change control, following this velocity will not lead the agent into a collision or violation of proximity constraints for the given horizon. If the set of valid velocities is empty, then the objective of maintaining coherence is dropped in favor of guaranteeing collision avoidance, which should be satisfiable, unless oscillations appear in the selected controls of agents. If two agents that are supposed to retain proximity end up violating the distance constraint, the proposed method makes them move in a direction that will allow them to reconnect.

This paper describes the LOCO method and provides simulations which show that, by reasoning about distance constraints, it is possible to solve decentralized collision avoidance problems while improving the coherence of a team. The approach is built on top of and compared against a formulation of velocity obstacles proposed in the literature for teams of agents that execute the same protocol, referred to as Reciprocal Velocity Obstacles [27, 22]. The experiments consider disk-shaped agents that move with a constant speed in a holonomic manner, i.e., the agents can freely choose to follow any direction instantaneously. First, a series of static environments are tested with a single team of agents navigating while maintaining coherence. Different types of formations between the teams are considered. Then, tests for multiple teams of agents navigating in the same environment while maintaining coherence are performed.

The organization of the paper is as follows. First, in Section 2, the relevant work to this problem is reviewed. Section 3 provides a formal description of the problem as well as the applicable notation used throughout the manuscript. The Velocity Obstacle framework is outlined in Section 4 together with the proposed approach for maintaining team coherence. Section 5 describes relevant results on various experimental setups. Lastly, Section 6 concludes the paper with a discussion of the technique and possible future work.

2. BACKGROUND

2.1 Virtual Agent Applications

The need to move multiple agents as a coherent team arises in many virtual agent applications [25, 18], ranging from crowd simulation [17], to pedestrian behavior analysis [16, 19], to shepherding and flocking behaviors [15, 29]. Many methods make use of "steering behaviors", with the objective of having agents navigate in a life-like and improvisational manner [20]. These steering behaviors can be combined to achieve higher-level goals, such as "get to the goal while avoiding obstacles" or "join a group of characters". Similar is the objective of the social force model [10].

2.2 Coupled Multi-Robot Path Planning

There is also extensive literature on motion coordination and collision avoidance in robotics. The multi-robot path planning problem can be approached by either a coupled approach or a decoupled one [12, 13]. The coupled approach plans for the composite robot, which has as many degrees of freedom as the sum of degrees of freedom of each individual robot. Integrated with complete/optimal planners, the coupled algorithm achieves completeness/optimality. Nevertheless, it becomes intractable due to its exponential dependency on the number of degrees of freedom.

2.3 Decoupled Multi-Robot Path Planning

Decoupled approaches plan for each agent individually. In prioritized schemes, paths are computed sequentially and high-priority agents are treated as moving obstacles by low-priority ones [6]. Searching the space of priorities can assist in performance [4]. Such decoupled planners tend to prune states in which higher priority agents allow lower priority ones to progress, which may eliminate the only viable solutions. Search-based decoupled approaches consider dynamic prioritization and windowed search [21], as well as spatial abstraction for improved multi-agent heuristic computation [24, 30]. In particular, mobile robotic sensor networks require that robots move while maintaining communication. Techniques which attempt to tackle this issue have to balance a trade-off between centralized and decentralized planning. There are techniques that create networks of robots to compute plans in a centralized manner across distributed systems [5] or in a fully decentralized manner [3].

2.4 Formations

There is a significant amount of work on formation control, which is a way of moving multiple agents as a coherent team. One direction is to use a virtual rigid body structure to define the shape of a formation, and then plan for this rigid body [14]. Other techniques attempt to have more flexible structures, where interactions between robots are modeled as flexible joints [1]. An alternative is to first generate a feasible trajectory for the group's leader according to its constraints and then use feedback controllers for the followers [7, 2].

2.5 Reactive Obstacle Avoidance

Many techniques attempt to solve the problem of collision avoidance using reactive methods. One technique used in robotics is the Dynamic Window approach, which operates directly in the velocity space of a robot, reasoning over the achievable velocities within a small time interval [9]. An

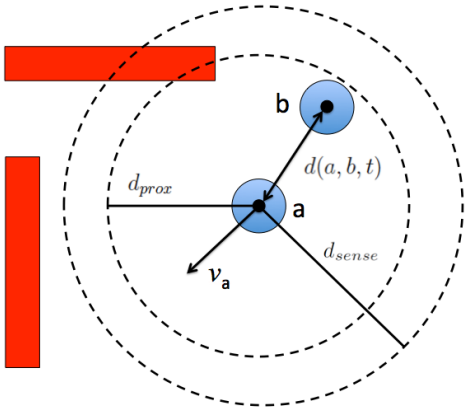


Figure 2: Agents a and b share a proximity constraint d_{prox}. Agent a moves with velocity v_a where $\|v\| = s$ and can sense all agents within d_{sense}.

alternative approach, known as the Velocity Obstacle (VO), assumes that neighboring agents will keep following their current control. Based on this assumption, it defines conic regions in velocity space, which are invalid to follow, as they lead to a collision with the neighbor at some time in the future [8]. If the future trajectory of other robots is known, non-linear VOs can be constructed [11]. The basic VO formulation can result in oscillations in motion when multiple agents execute the same algorithm. The reciprocal nature of other robots can be taken into account in order to avoid these oscillations, which leads to the definition of Reciprocal Velocity Obstacles [27]. Using this idea of reciprocity, the robots can attempt to optimally steer out of collision courses with other robots using an extension of this framework called Optimal Reciprocal Collision Avoidance (ORCA) [26]. This technique was extended to 3D cases using simple-airplane systems [23]. Further work extends the VO formulation to work with acceleration constraints as well as many kinematically and dynamically constrained systems [28].

2.6 Contribution

This work uses a reactive technique to define new obstacles in the velocity space, called Loss of Communication Obstacles (LOCO). LOCOs are computed quickly and, when integrated with Velocity Obstacles, allow robots to reactively avoid static and dynamic obstacles while maintaining a better sense of coherence. The new technique does not impose communication requirements, yet maintains connectivity in a decentralized manner. The approach still requires that robots are able to sense the position and velocity of other robots in the scene, as well as the position of static obstacles.

3. PROBLEM SETUP

Consider n planar, holonomic disks moving with constant speed s. Let the set of all agents be A. Each disk agent $a \in A$ has radius r_a and can instantaneously move with a velocity vector v_a that has magnitude s. Agents are assumed to be capable of sensing the position and velocity of other agents in the environment within a sensing radius d_{sense}. Furthermore, the agents have available a map M of the environment that includes the static obstacles. The configuration space for each agent is $Q = \mathbb{R}^2$, and it can be partitioned into two sets, Q_{free} and Q_{obst}, where Q_{free} represents the obstacle free part of the space, and Q_{obst} is the part of the space with obstacles. Each agent follows a trajectory $q_a(t)$, where t is time. Initially an agent is located at a configuration $q_a(0) = q_a^{init}$ and has a goal location q_a^{goal}.

Consider the distance $d(a, b, t)$ between agents a and b at time t (Figure 2). If $d(a, b, t) < r_a + r_b$, then agents a and b are said to be in collision at time t. Collisions with static obstacles occur when $q_a(t) \in Q_{obst}$. An input graph $G(A, E)$ is provided that specifies which agents need to be in close proximity. The vertices of graph G correspond to the set of agents A and an edge (a, b) implies that agents a and b must satisfy $d(a, b, t) \leq d_{prox}$, where d_{prox} is a proximity constraint.

The objective is for the agents to move from q^{init} to q^{goal} without any collisions with obstacles or among them, while satisfying as much as possible the proximity constraints specified in the input graph G. More formally, all agents should follow trajectories $q_a(t)$ for $0 \leq t \leq t_{final}$, so that $\forall a \in A$:

- $q_a(0) = q_a^{init}$,
- $q_a(t_{final}) = q_a^{goal}$,
- $q_a(t) \in Q_{free}$, $\forall t \in [0, t_{final}]$,
- and $\forall b : r_a + r_b < d(a, b, t) < D$, where
  - $D = d_{prox}$, if $\exists (a, b) \in E$ of graph G,
  - and $D = \infty$ otherwise.

4. APPROACH

4.1 Velocity Obstacle Framework

Velocity obstacles are defined in the relative velocity space of two agents. $VO^{\infty}_{a|b}$ can be geometrically constructed as in Figure 3. Agent a's geometry is reduced to a point by performing the Minkowski sum of agent a and agent b: $a \oplus b$. Then, tangent lines to the Minkowski sum disk $a \oplus b$ are constructed from agent a. These tangent lines bound a conic region. This region represents the space of all velocities v_a of agent a that would eventually lead into collisions with agent b assuming that b has zero velocity. Given that agent b has a velocity v_b, the conic region needs to be translated by the vector v_b. This construction assumes an infinite time horizon. In practice, it is often helpful to truncate the VO based on a finite time horizon τ. This VO represents all velocities v_a, which will lead into a collision within time $t \leq \tau$, given that agent b keeps moving with velocity v_b.

This work adopts a modification of the basic VO framework, which deals with the case that the two agents are reciprocating [27]. Reciprocal Velocity Obstacles (RVOs) are created by translating the VO according to a weighted average of the agents' velocities, $(\alpha \cdot v_A) + ((1 - \alpha) \cdot v_B)$ where α is a parameter representing the level of reciprocity between agents A and B. If both agents are equally reciprocal, then $\alpha = 0.5$.

Given this framework, an algorithm for calculating valid velocities for decentralized collision avoidance can be defined. Agent a will have a set of reachable velocities and the task is to select one such velocity, which is collision-free. For holonomic disk agents moving at constant speed s, the set of reachable velocities would be $V_{reach} = \{v | \|v\| = s\}$. Let the set of velocities which are invalid according to all the VOs for neighbors of a be defined as follows: $V_{inv} = \{v | \exists\ \mathtt{VO}_{a|b}\ s.t.\ v \in \mathtt{VO}_{a|b}, b \in A, a \neq b\}$. Then, the set of feasible velocities which are reachable but not in the invalid set is defined as $V_{feas} = V_{reach} \setminus V_{inv}$. The selected veloc-

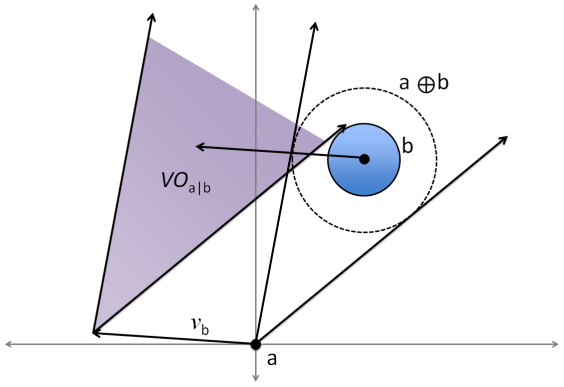


Figure 3: Construction of $VO^{\infty}_{a|b}$ for an infinite time horizon. The Minkowski sum of a and b, $a \oplus b$ is used to define a cone in velocity space, which is then translated by v_b. The shaded region represents all velocities v_a, which lead a into collision with b.

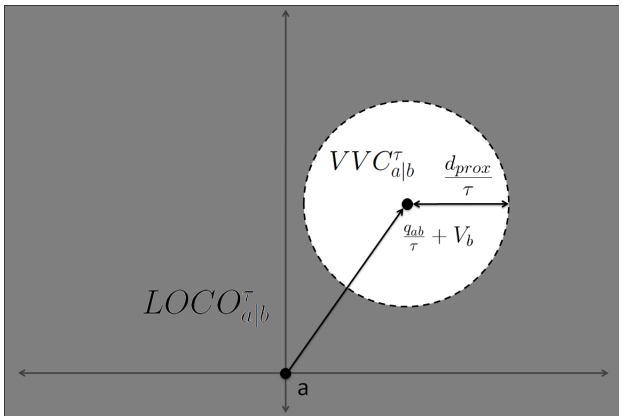


Figure 4: Construction of $\texttt{LOCO}^{\tau}_{a|b}$ and $\texttt{VVC}^{\tau}_{a|b}$. The circular region represents the viable velocities to maintain $d(a,b,t) \leq d_{prox}$ for time horizon τ. The shaded region outside the disk represents invalid velocities for agent a.

ity should be feasible, $v \in V_{feas}$, and typically minimizes a metric relative to the preferred velocity v_a^{pref}, e.g., a velocity vector that points to the goal. More details will be provided regarding the specific velocity selection scheme used in this work, in section 4.4.

4.2 Loss of Communication Obstacles

The proposed approach extends the VO framework by creating new obstacles in the velocity space. These obstacles aim to prevent loss of communication, or more generally, to satisfy proximity constraints between agents in the form $d(a,b,t) \leq d_{prox}$ for agents a and b for at least a finite time horizon τ. The LOCO imposed by agent b on agent a for a horizon τ, denoted as $\texttt{LOCO}^{\tau}_{a|b}$, is the set

$$\texttt{LOCO}^{\tau}_{a|b} = \{v | \ \forall \ t \in [0,\tau] : d(a,b,t) \leq d_{prox}\},$$

under the assumption that agent b follows its current velocity v_b for at least time τ.

Assume agents a, $b \in A$ for which $(a,b) \in E$ of the input graph G. The relative position of agent b for agent a will be denoted as $q_{ab} = q_b - q_a$. If the relative position at time t is $q_{(ab)}(t)$, then at time $t+\tau$ the relative position of the two agents is going to be:

$$q_{ab}(t+\tau) = q_{(ab)}(t) + \tau * (V_b - V_a).$$

For the two agents to be able to communicate at time $t+\tau$, it has to be that:

$$(q^X_{ab}(t+\tau))^2 + (q^Y_{ab}(t+\tau))^2 \leq d^2_{prox} \Rightarrow$$

$$(q^X_{ab}(t)+\tau*(V^X_b - V^X_a))^2 + (q^Y_{ab}(t)+\tau*(V^Y_b - V^Y_a))^2 \leq d^2_{prox} \Rightarrow$$

$$(\frac{q^X_{ab}(t)}{\tau} + V^X_b - V^X_a)^2 + (\frac{q^Y_{ab}(t)}{\tau} + V^Y_b - V^Y_a)^2 \leq \frac{{d^{prox}}^2}{\tau^2} \Rightarrow$$

$$(V^X_a - \frac{q^X_{ab}(t)}{\tau} - V^X_b)^2 + (V^Y_a - \frac{q^Y_{ab}(t)}{\tau} - V^Y_b)^2 \leq (\frac{d_{prox}}{\tau})^2$$

The last expression implies that the velocity V_a of agent a has to be within a circle with center $(\frac{q_{ab}}{\tau} + V_b)$ and radius $\frac{d_{prox}}{\tau}$. This circle will be referred to as the valid velocity circle ($\texttt{VVC}^{\tau}_{a|b}$) and is the complement of the $\texttt{LOCO}^{\tau}_{a|b}$. Figure 4 gives an example of a VVC circle.

An agent a, however, may have multiple neighbors leading to the definition of multiple LOCOs and VVCs. A choice that is made for simplicity is to consider the same time horizon τ for the definition of all the LOCOs for all the neighbors. Then, the set of velocities v_a that will not allow a to maintain connectivity with at least one neighbor b in the graph G is the union of individual LOCOs:

$$\texttt{LOCO}^{\tau}_{a} = \bigcup_{\forall b \ | \ \exists (a,b) \in E} \texttt{LOCO}^{\tau}_{a|b}$$

There are two complications arising from this definition. Firstly, it is not straightforward to compute the longest horizon for which this union is not the entire plane, i.e., the longest horizon for which the intersection of VVCs is not empty. The problem is that both the centers and the radii of the VVCs are changing for different time horizons. Secondly, the resulting region is rather complex to describe, as it corresponds to multiple circle intersections. This representation can impose significant computational overhead when the LOCOs are integrated with VOs.

Instead of computing the exact intersection of VVCs, this paper proposes a conservative approximation that is easier to represent and beneficial for computational purposes. The approximation of the valid set of velocities corresponds to a circle inside the intersection of VVCs. Figure 5 illustrates the procedure for two and three neighbors. Given two circles with centers C_b and C_c and radii r_b and r_c, the inscribed circle of their intersection has the following radius and center:

$$r_{bc} = \frac{r_b + r_c - ||C_b, C_c||}{2}$$

$$C_{bc} = C_b + (r_b - r_{bc})\frac{C_c - C_b}{||C_b, C_c||}$$

The procedure works in an incremental manner. First it computes the inscribed circle (C_{bc}, r_{bc}) of the intersection of two VVCs, and then computes the inscribed circle of (C_{bc}, r_{bc}) with another VVC and so on.

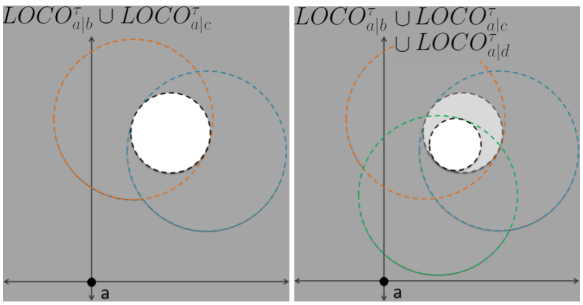


Figure 5: The conservative approximation of $\texttt{VVC}^{\tau}_{a}$ in the velocity space of agent a for two (left) and three (right) neighbors. The white circle corresponds to velocities that are guaranteed to maintain connectivity with the neighbors for time τ.

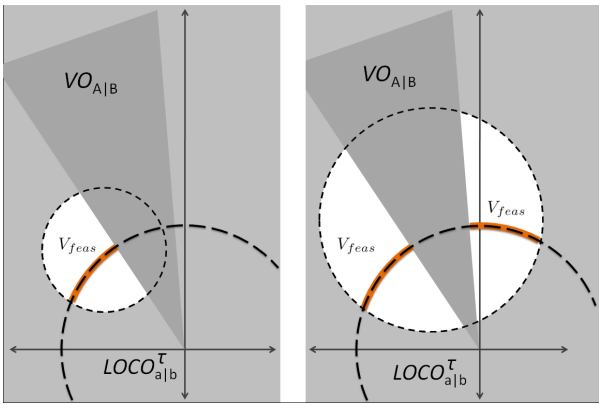


Figure 6: An example of how changing the horizon affects the set of feasible controls. The left image has a larger value for τ while the right has a smaller value of τ. Larger values of τ provide stronger guarantees for communication maintenance, but make finding a feasible control more difficult.

4.3 Integration of VOs and LOCOs

The final step for computing the $\texttt{LOCO}^{\tau}_{a}$ is to select a suitable time horizon. A tuning approach over consecutive simulation steps is used. Given the horizon τ from the previous time step, the $\texttt{LOCO}^{\tau}_{a}$ is computed as in Figure 5. Then the set $V_{valid} = V_{reach} \setminus \texttt{LOCO}^{\tau}_{a}$ is computed, which considers the reachable controls that do not violate the LOCO constraints. This operation can be done in an efficient manner especially for systems with constant speed s, as it corresponds to a circle to circle intersection. In the general case for systems with varying velocity there are still computational advantages, as the circular representation of the VVC greatly increases the speed in which V_{valid} can be computed. Once V_{valid} is available for a given τ, the measure of $|V_{valid}|$ is compared against a predefined threshold $|V_{thresh}|$. If $|V_{valid}| < |V_{thresh}|$, then τ is decreased and V_{valid} is recomputed. A smaller τ implies that a bigger set of valid velocities for proximity maintenance will be returned that provides guarantees for a shorter horizon. Alternatively, if $|V_{valid}| > |V_{thresh}|$, then τ is increased. This will return a smaller set of valid velocities but will provide guarantees for a longer horizon. This tuning process continues until V_{valid} comes close to V_{thresh}, but this takes place over multiple simulation steps and adapts on the fly to changes in the relative configuration of agents. The effects of tuning τ can be seen in Figure 6.

Once $\texttt{LOCO}^{\tau}_{a}$ has been computed, it must then be included in the list of constraints in order to correctly compute V_{feas}, which is now redefined to be $V_{feas} = V_{valid} \setminus V_{inv}$, as in Figure 6. Sometimes, the additional constraints imposed by LOCOs, cause V_{feas} to become empty. In this situation, the LOCO constraints are ignored, as attempting to maintain communication may be perilous to the agent's safety.

4.4 Velocity Selection

Each agent has a preferred velocity it would like to follow, denoted as v^{pref}_{a} for agent a. Ignoring the proximity constraints and in obstacle-free environment, the preferred velocity should be in the direction of the goal configuration $v_{goal} = q^{goal}_{a} - q_a$. If there are obstacles, however, setting the goal velocity in the same manner can lead the agents into local minima, causing the agent to become stuck behind the obstacle. This work avoids this issue by computing a discrete wave-front function in environments with obstacles. The goal velocity is computed as the direction to the cell with the minimum distance to the goal within a 3×3 region from the current cell of the agent.

In order to account for agents violating proximity constraints, a weighted velocity selection scheme is employed that takes neighbors into account. For some agent a, let d_i be the distance from agent a to another agent i and X_i be the state of agent i, where agent i is part of the proximity graph of agent a. Then, for n agents in the proximity graph of agent a, the average weighted configuration can be computed as:

$$q_{avg} = \frac{\sum_i^n \frac{d_i}{d_{prox}} q_i}{\sum_i^n \frac{d_i}{d_{prox}}}$$

Then, $d_{avg} = ||q_a - q_{avg}||$ is the distance between agent a and q_{avg}. Let $v_{avg} = q_{avg} - q_a$ be the vector pointing to q_{avg}, and let v_{goal} be the vector towards the goal computed through the wavefront. Then, agent a's preferred velocity is computed as follows:

$$v^{pref}_{a} = \frac{d_{avg}}{d_{prox}} v_{avg} + \frac{d_{prox} - d_{avg}}{d_{prox}} v_{goal}$$

Thus, agents which are farther away from their proximity constraints (i.e., have violated constraints) will be inclined to shorten this distance, whereas agents that have not violated any proximity constraints move towards their goals.

Once v^{pref}_{a} is computed, it can be checked for validity given the set V_{feas}. If v^{pref}_{a} is feasible, then agent a will use it. In the case that v^{pref}_{a} is not in V_{feas}, a different feasible control must be computed. In the general case, the region V_{feas} defines an area in velocity space which must be searched to find a control. The control found is the velocity $v \in V^{a}_{feas}$ which minimizes distance between v and v^{pref}_{a}.

The distance metric used in this approach is the Euclidean distance in the velocity space. Other metrics for finding the distance between velocities in this space are possible [28]. A list of intersections of the boundaries of the RVOs with V^{a}_{reach} is generated and a rotational sweep algorithm determines which points are valid. These points define the boundaries of the V^{a}_{feas} regions. They are checked to find which one of them minimizes the distance to the preferred velocity :

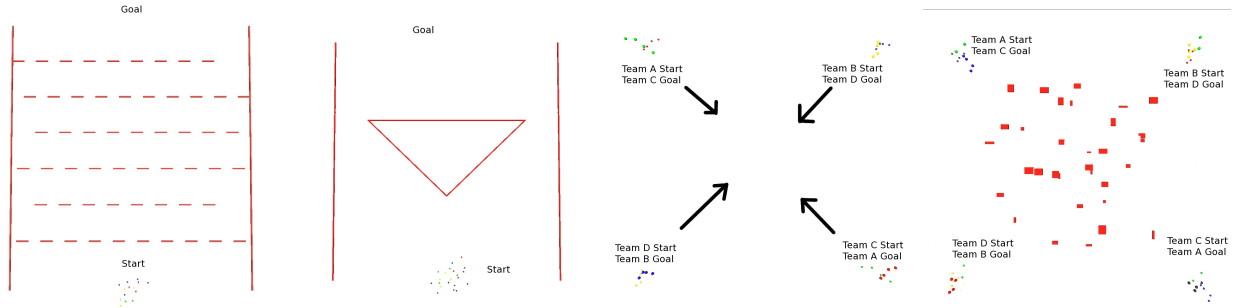


Figure 7: The environments on which the experiments were executed

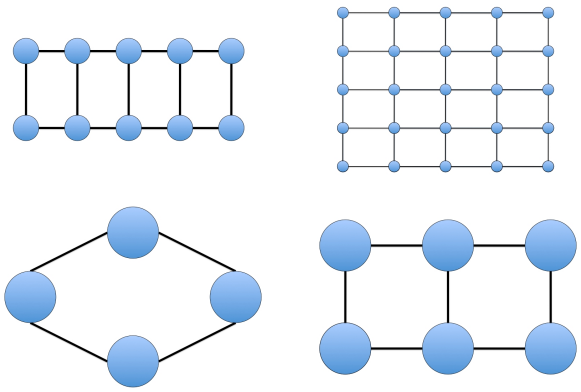


Figure 8: Examples of input proximity graphs used in the experiments.

$min_{v \in V^a_{feas}}[d(v, v^{pref}_a)]$. In the general case, V^a_{feas} will be a non-convex region or possibly disjoint non-convex regions in the velocity space where a variant of the simplex algorithm can be used to find the optimum.

In the event that there is no valid velocity available, the agent will select its current velocity. The reasoning behind this is that in the VO framework, agents assume that their neighbors will keep using their current control. Thus, choices that keep the current control are preferable for this scheme.

5. RESULTS

The approach was implemented using a simulation software platform. Experiments were run on computers with a 3.06 GHz Intel Core 2 Duo processor and 4GB of RAM. The experiments are organized in the following manner. First, experiments were conducted using a single team of agents moving in an environment with obstacles. Then, experiments with multiple teams were run both in an obstacle-free world and in an environment with obstacles. A team of agents corresponds to a connected component of the input graph G. The experiments conducted compare the `LOCO` formulation against `RVO`s. The reason for comparing against `RVO`s, rather than other formation techniques, is due to the decentralized nature of the `LOCO` algorithm, which requires no additional information outside the `VO` framework. In addition, `LOCO`s utilize the notion of coherence, which is more abstract in nature than formations. Thus, `RVO`s are the most relevant technique to experiment against. Each experiment was measured with regards to the following metrics:

- total number of collisions during the entire experiment,
- computation time per frame,
- total time to solve the problem,
- average ratio of respected proximity links per frame,
- and number of successful runs.

For each variation of the environment, input graph G and algorithm, there were 10 runs executed. Each run had a random initial q_{init} and goal configuration q_{goal}, which, satisfied the proximity link in the graph G.

5.1 Single Team

For the single team scenarios, two different environments were tested with varying numbers of agents. The `PACHINKO` environment uses a series of walls with small gaps (Figure 7(left)), and a team of 10 agents was considered in this case. The proximity graph imposed on the team is shown in Figure 8(top left). The `WEDGE` environment (Figure 7(middle left)) attempts to split the agents into two lanes. The proximity graph imposed on the team is shown in Figure 8(top right).

	time per step (s)		collisions		maintained links		steps		success	
	LOCO	RVO	LOCO	RVO	LOCO	RVO	LOCO	RVO	LOCO	RVO
wedge grid	0.03964	0.03488	0.1	0	92%	65%	4279.1	3376.9	100%	100%
pachinko train	0.04474	0.0355	0	0	96%	46%	3881.2	3555	100%	100%

Table 1: Results for a single team.

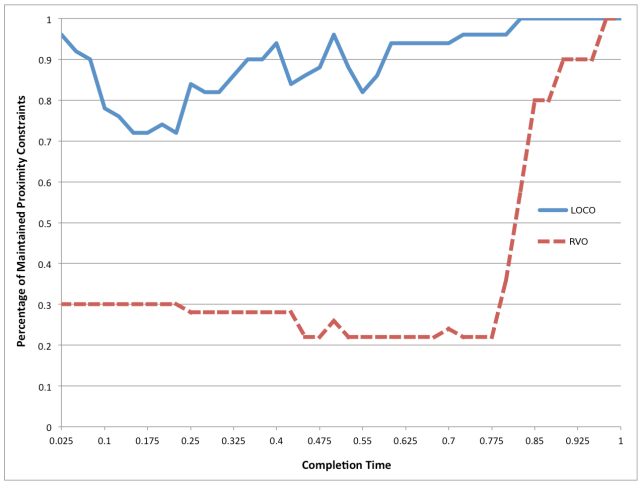


Figure 9: Coherence over time for 24 agents in the wedge grid experiment.

Table 1 shows that there is a significant improvement in terms of the percentage of links maintained by the LOCO-based approach relative to RVOs. Another interesting statistic to examine is the coherence of agents over time, shown in Figure 9, which the LOCO-based approach also improves. This comes at the cost of a slightly increased computational cost and increased time taken by the algorithm to bring all of the agents to their goals. An example of the paths taken by 24 agents in the PACHINKO environment is shown in Figure 10. Both approaches were able to solve all of the problems and without any collision, with the exception of one experiment for the WEDGE environment, where a single collision was reported.

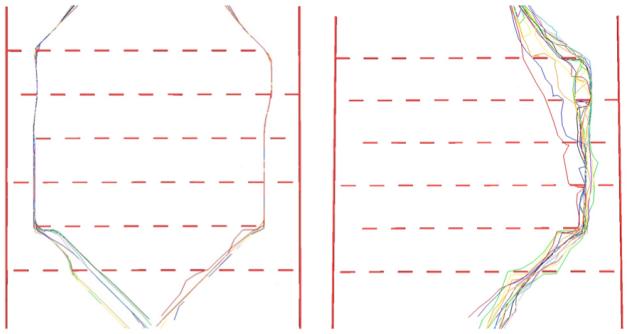


Figure 10: An example of the paths taken by 24 agents in the PACHINKO environment for the RVO (left) and LOCO (right) approach.

5.2 Multiple Teams

The performance of LOCOs was evaluated against RVOs in a scenario involving four teams of agents navigating in an obstacle-free environment as shown in Figure 7(middle right) (CROSSROADS) and for the connectivity graph for each team shown in Figure 8(bottom left). There were four agents in each team. The last setup involved again four teams of agents, each one of which had six agents connected as in Figure 8(bottom right). This time the environment involved a random set of rectangular obstacles as in Figure 7(right). For each of the 10 runs the rectangles were randomly placed. The results are shown in Table 2.

In both the random and crossroads examples, LOCOs outperform RVOs as far as connections are concerned, though oftentimes LOCOs take more time to solve the specified problem. On different problems, both approaches can take some extra time to complete the problem for two different reasons. The VOs take extra time as agents will sometimes get caught on obstacles or are pushed away from their goals by agents on other teams. LOCOs may take extra time as they spend effort navigating agents in densely-packed situations that occur at the center of the environment as they cross over. Furthermore, LOCOs exhibit a flocking behavior which sometimes causes agents to overshoot their goals.

	time per step (s)		collisions		maintained links		steps		success	
	LOCO	RVO	LOCO	RVO	LOCO	RVO	LOCO	RVO	LOCO	RVO
random	0.03005	0.02919	0	0.1	96%	72%	2840.78	2416.3	90%	100%
crossroads	0.00564	0.0034	0	0	98%	71%	2871.1	2596	100%	100%

Table 2: Results for multiple teams.

In the random obstacle environment, LOCOs imposed a very small computational overhead. The crossroads scenario resulted in a more significant increase, because all agents meet in the center of the environment nearly at the same point in time. This increases computational cost, as LOCOs must tune their horizon and reconnect broken proximities more frequently. In both environments, LOCOs and RVOs resulted in no collisions, with the exception of one run of RVOs in the random environment. The main focus of these results, however, is the improvement in maintained links. The LOCO-based approach maintained 95% of the proximity links, which was an improvement over RVOs. LOCOs cause a small degradation in path quality, which is measured in the number of steps it took for agents to solve the environment. In addition to this, LOCO failed to solve one of the randomly generated environments, because agents tend to spread to the edge of their proximity constraints, which may cause problems when agents move around obstacles. Overall, the results seem to validate the initial approach, as the goal of LOCOs is to maintain safety while providing better proximity maintenance is experimentally supported.

6. DISCUSSION

This work provides the formal definition for a new kind of obstacle in the velocity space of moving agents, referred to as a Loss of Communication Obstacle (LOCO), which correspond to proximity constraints with neighboring agents. These obstacles can be easily computed and integrated into the existing framework of Velocity Obstacles for decentralized collision avoidance. Additionally, an approach for tuning the time horizon parameter for these obstacles over multiple simulation steps is provided. These additional constraints increase the overall coherence of teams of agents while navigating through environments with static obstacles and other moving bodies. LOCOs can be dropped if it is determined that it is too difficult to maintain proximity without jeopardizing safety. The implementation of the technique shows improved coherence for agents who share communication links without sacrificing safety at a small computational overhead.

A natural extension is to consider more challenging systems, including kinematically and dynamically constrained systems. Adapting LOCOs to work in these cases is possible, as there have already been extensions on using VOs with dynamics. Since these extensions are in an orthogonal direction to the LOCO algorithm, it is easy so as to achieve decentralized coherence maintenance for dynamic systems. Further extending the work to applications of real robots will require introducing a method for handling sensor errors, as well creating a more robust reconnection strategy built on this error model. One drawback of reactive techniques is that they may get stuck in local minima. It is interesting to better study the integration with the wavefront approach or another global planner so as to guarantee that agents will make progress towards their goal while guaranteeing safety and connectivity. A global planner can also improve the performance of the reconnection strategy. Currently, the direction to the agent with which connectivity has been lost ignores the existence of local obstacles. Another interesting direction is to study reciprocity tuning. More constrained agents, such as those who create a bridge for two otherwise disconnected agents, may be less able to reciprocate than others. Further reducing the computational overhead of the technique would also have great benefits, since larger-scale

tests could be performed and have practical application areas, as in crowd simulation and video games.

7. REFERENCES

[1] Balch, T. and Hybinette, M. Social potentials for scalable multi-robot formations. In *Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on*, volume 1, pages 73–80, 2000.

[2] T. D. Barfoot and C. M. Clark. Motion planning for formations of mobile robots. *Journal of Robotics and Autonomous Systems*, 46(2), Feb. 2004.

[3] K. E. Bekris, K. I. Tsianos, and L. E. Kavraki. Safe and distributed kinodynamic replanning for vehicular networks. *Mobile Networks and Applications*, 14(3), February 2009.

[4] M. Bennewitz, W. Burgard, and S. Thrun. Finding and Optimizing Solvable Priority Schemes for Decoupled Path Planning Techniques for Teams of Mobile Robots. *Robotics and Autonomous Systems*, 41(2):89–99, 2002.

[5] C. M. Clark, S. M. Rock, and J.-C. Latombe. Motion Planning for Multiple Robots using Dynamic Networks. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 4222–4227, 2003.

[6] M. Erdmann and T. Lozano-Perez. On multiple moving objects. In *IEEE Intern. Conference on Robotics and Automation (ICRA)*, pages 1419–1424, 1986.

[7] R. Fierro, C. Belta, J. Desai, and V. Kumar. On controlling aircraft formations. In *Decision and Control, 2001. Proceedings of the 40th IEEE Conference on*, volume 2, pages 1065 –1070 vol.2, 2001.

[8] P. Fiorini and Z. Shiller. Motion planning in dynamic environments using velocity obstacles. *Int. Journal of Robotics Research*, 17(7), 1998.

[9] D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Robotics and Automation Magazine*, 4(1), 1997.

[10] D. Helbing, L. Buzna, A. Johansson, and T. Werner. Self-organized pedestrian crowd dynamics: Experiments, simulations and design solutions. *Transportation Science*, 2005.

[11] F. Large, C. Laugier, and Z. Shiller. Navigation among moving obstacles using the NLVO: Principles and applications to intelligent vehicles. *Autonomous Robots*, 19(2), 2005.

[12] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[13] S. M. LaValle. *Planning Algorithms*. Cambridge, 2006.

[14] M. A. Lewis and K.-H. Tan. High precision formation control of mobile robots using virtual structures. *Auton. Robots*, 4:387–403, October 1997.

[15] J. M. Lien, S. Rodriguez, J. P. Malric, and N. Amato. Shepherding behaviors with multiple shepherds. In *Intern. Conf. on Robotic and Automation*, 2005.

[16] S. Paris, J. Pettre, and S. Donikian. Pedestrian reactive navigation for crowd simulation: A predictive approach. *Computer Graphics Forum*, September 2007.

[17] N. Pelechano, J. Allbeck, and N. Badler. Controlling individual agents in high-density crowd simulation. In *ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SCA)*, volume 3, pages 99–108, San Diego, CA, 2007.

[18] N. Pelechano, J. Allbeck, and N. Badler. *Virtual Crowds: Methods, Simulation and Control*. Morgan and Claypool Publishers, 2008.

[19] J. Pettre, J. Ondrej, A.-H. Olivier, A. Cretual, and S. Donikian. Experiment-based modeling, simulation and validation of interactions between virtual walkers. In *Symposium on Computer Animation*. ACM, 2009.

[20] C. W. Reynolds. Steering behaviors for autonomous characters. In *Proc. of the Game Developers Conference (GDC)*, pages 763–782, San Jose, CA, 1999.

[21] D. Silver. Cooperative pathfinding. In *The 1st Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE'05)*, pages 23–28, 2005.

[22] J. Snape, S. J. Guy, J. van den Berg, S. Curtis, S. Patil, M. C. Lin, and D. Manocha. Independent navigation of multiple robots and virtual agents. In *Proc. of the 9th Int. Conf. on Autonomous Agents and Multiagents Systems (AAMAS 2010)*, Toronto, Canada, May 2010.

[23] J. Snape and D. Manocha. Navigating multiple simple-airplanes in 3d workspace. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2010.

[24] N. Sturtevant and M. Buro. Improving collaborative pathfinding using map abstraction. In *The Second Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE'06)*, pages 80–85, 2006.

[25] D. Thalmann. Populating virtual environments with crowds. In *Int. Conf. on Virtual Reality Continuum and Its Applications*. ACM, 2006.

[26] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha. Reciprocal n-body collision avoidance. In *Proc. Int. Symposium of Robotics Research*. International Foundation on Robotics Research, aug. 2009.

[27] J. Van den Berg, M. Lin, and D. Manocha. Reciprocal velocity obstacles for real-time multi-agent navigation. In *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2008.

[28] J. Van den Berg, J. Snape, S. Guy, and D. Manocha. Reciprocal Collision Avoidance with Acceleration-Velocity Obstacles. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, May 2011.

[29] C. Vo, J. F. Harrisong, and J. M. Lien. Behavior-based motion planning for group control. In *Intern. Conf. on Intelligent Robots and Systems*, St. Louis, Mo, 2009.

[30] K.-H. C. Wang and A. Botea. Fast and memory-efficient multi-agent pathfinding. In *International Conference on Automated Planning and Scheduling (ICAPS)*, pages 380–387, Sydney, Australia, 2008.

Session 2B
Distributed Problem Solving

Stochastic Dominance in Stochastic DCOPs for Risk-Sensitive Applications

Duc Thien Nguyen
School of Information Systems
Singapore Management University
Singapore 178902
dtnguyen@smu.edu.sg

William Yeoh
School of Information Systems
Singapore Management University
Singapore 178902
williamyeoh@smu.edu.sg

Hoong Chuin Lau
School of Information Systems
Singapore Management University
Singapore 178902
hclau@smu.edu.sg

ABSTRACT

Distributed constraint optimization problems (DCOPs) are well-suited for modeling multi-agent coordination problems where the primary interactions are between local subsets of agents. However, one limitation of DCOPs is the assumption that the constraint rewards are without uncertainty. Researchers have thus extended DCOPs to Stochastic DCOPs (SDCOPs), where rewards are sampled from known probability distribution reward functions, and introduced algorithms to find solutions with the largest expected reward. Unfortunately, such a solution might be very *risky*, that is, very likely to result in a poor reward. Thus, in this paper, we make three contributions: (1) we propose a stricter objective for SDCOPs, namely to find a solution with the most stochastically dominating probability distribution reward function; (2) we introduce an algorithm to find such solutions; and (3) we show that stochastically dominating solutions can indeed be less risky than expected reward maximizing solutions.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed AI

General Terms

Algorithms, Experimentation

Keywords

DCOP, DPOP, Uncertainty, Stochastic Dominance

1. INTRODUCTION

Distributed constraint optimization problems (DCOPs) are problems where agents need to coordinate their value assignments to maximize the sum of the resulting constraint rewards. They are well-suited for modeling multi-agent coordination problems where the primary interactions are between local subsets of agents, such as the scheduling of meetings [15], the coordination of sensors in networks [5, 14], the coordination of first responders in disasters [11], the management of power plant networks [10] and the generation of coalition structures [24].

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

One of the limitations of DCOPs is that they only model problems with (constraint) rewards that are both known a priori and without uncertainty. Thus, researchers have introduced various DCOP extensions to address this limitation. One example is the Distributed Coordination of Exploitation and Exploration (DCEE) model [8, 23], where agents initially do not know the constraint rewards and can only discover them through exploration. The agents in a DCEE problem coordinate to balance exploration and exploitation such that they accumulate the maximum amount of reward over a finite time horizon. Another example is the DCOP under Stochastic Uncertainty model [12], where there are random variables that take on values according to known probability distribution functions. The values of these random variables affect the overall reward. Finally, researchers have introduced a Stochastic DCOP (SDCOP) model where the constraint rewards are no longer deterministic values but are sampled from known probability distribution functions called reward functions [1]. Agents in these latter two problems coordinate to maximize the expected reward of the overall solution.

Finding a solution that maximizes the expected reward in an SDCOP can be done in a straightforward manner if the means of the probability distribution functions are known. Since the sum of the expected reward of two functions equals the expected reward of the convolution of the two functions according to the linearity of expectations, one can map an SDCOP to a DCOP by mapping the reward function in an SDCOP to the mean of that function in a DCOP. One can then use existing DCOP algorithms to solve the problem.

Our concern with the approach above is that a solution with the maximum expected reward might be very *risky*, that is, very likely to result in a poor reward, particularly in high variance environments. As an example, one might prefer to have a solution whose overall reward function follows a unimodal distribution rather than a bimodal distribution especially if the expected reward of the unimodal distribution is only slightly smaller than that of the bimodal distribution. Thus, in this paper, we propose a stricter objective for SDCOPs, namely to find a solution with the most stochastically dominating probability distribution function of the reward of the solution [6]. Intuitively, a stochastically dominating function is less risky than the dominated function.

We also introduce Stochastic Dominance DPOP (SD-DPOP), an extension of DPOP [18], to solve SDCOPs. Our results show that stochastic dominating solutions found by SD-DPOP are less risky compared to expected reward maximizing solutions found by DPOP for common risk functions.

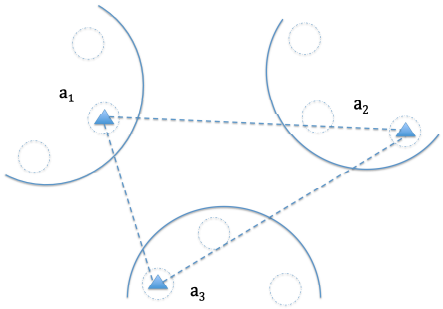


Figure 1: Illustration of Mobile Agents

2. MOTIVATING DOMAIN

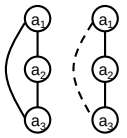
First responders in an earthquake hit area or soldiers in the battlefield might need to deploy mobile agents to create a mesh of communication network to coordinate their actions [16]. Therefore, we motivate our work in this paper with the problem of maximizing the signal strengths between neighboring mobile agents in a network [8]. Figure 1 shows an illustration of three mobile agents, whose positions are represented by triangles. Straight dotted lines connect neighboring agents. Each mobile agent can choose to be in one of three possible nearby locations, which are denoted by circular dotted lines. We assume that the mobile agents can only move within a small range from their starting locations and the topology of the network thus remains unchanged. The signal strength between two neighboring agents depends on their locations. For example, the further their distance, the weaker the signal strength. The small movements of the mobile agents can affect the signal strength significantly due to radio interference. However, the signal strength between a pair of neighboring agents can be modeled as an independent random number drawn from some distribution [9]. In this paper, we will use Gaussian functions to model the signal strengths. The objective in this problem is for the agents to coordinate their choice of locations such that the sum of the signal strength between every pair of neighboring agents is maximized. We use these two domains as our motivating domains because they are examples that call for one to be risk-averse. In both domains, losing the ability to communicate can result in the loss of lives.

3. BACKGROUND

In this section, we provide a brief description of the DCOP and Stochastic DCOP model as well as the DPOP algorithm.

3.1 DCOPs

A *distributed constraint optimization problem* (DCOP) [17, 18] is defined as a tuple $\langle A, X, D, F\rangle$. $A = \{a_i\}_0^n$ is the finite set of agents. $X = \{x_i\}_0^n$ is the set of variables, where x_i is owned by agent a_i. $D = \{d_i\}_0^n$ is the set of finite domains, where domain d_i is the set of possible values for agent $a_i \in A$. $F = \{f_i\}_0^m$ is the set of binary constraints, where each constraint $f_i : d_{i_1} \times d_{i_2} \to \mathbb{R}^+ \cup \{0\}$ specifies its non-negative reward as a function of the values of the two different variables $x_{i_1}, x_{i_2} \in X$ that share the constraint. Although the general DCOP definition allows one agent to own multiple variables as well as the existence of n-ary constraints, we restrict our definition here for simplification purposes. One can transform a general DCOP to our DCOP using pre-processing techniques [2]. A solution is a value assignment for a subset of variables. Its reward is the sum of the constraint rewards of all constraints shared by variables with assigned values. A solution is *complete* iff it is a value assignment for all variables. Solving a DCOP optimally means finding a reward-maximal complete solution.



(a) (b)

for $i < j$

x_i	x_j	Rewards	Reward Functions
0	0	5	$\mathcal{N}(5,1)$
0	1	8	$\mathcal{N}(8,1)$
1	0	20	$\mathcal{N}(20,1)$
1	1	3	$\mathcal{N}(3,1)$

(c)

Figure 2: Example DCOP

DCOPs are commonly visualized as *constraint graphs*, whose vertices are the agents and whose edges are the constraints. Most complete DCOP algorithms operate on *pseudo-trees*, which are spanning trees of fully connected constraint graphs such that no two vertices in different subtrees of the spanning tree are connected by an edge in the constraint graph. Figure 2(a) shows the constraint graph of an example DCOP with three agents controlling variables that can each be assigned the values 0 or 1, Figure 2(b) shows one possible pseudo-tree (the dotted line is called a *backedge*, which is an edge of the constraint graph that does not connect a pair of parent-child nodes), and the third column in Figure 2(c) shows the constraint rewards. We will use this problem as a running example in this paper.

3.2 Stochastic DCOPs

Stochastic DCOPs (SDCOPs) are extensions of DCOPs where each constraint $f_i : d_{i_1} \times d_{i_2} \to P(x_{i_1}, x_{i_2})$ now specifies a (potentially discretized) probability distribution function of the reward as a function of the values of the two different variables $x_{i_1}, x_{i_2} \in X$ that share the constraint [1]. We call these functions *reward functions* in this paper. For example, the fourth column in Figure 2(c) shows the Gaussian reward functions for the different pairs of values. We assume that the reward functions are all independent of each other. Solving an SDCOP optimally means finding a complete solution X^* that maximizes the expected sum of all rewards:

$$X^* = \underset{X \in d_1 \times d_2 \times \cdots \times d_n}{\arg\max} \left\{ \mathbb{E}\left[\sum_i f_i(x_{i_1}, x_{i_2} \mid x_{i_j} \in X) \right] \right\} \quad (1)$$

One can solve for the this objective in a straightforward manner if the mean of each reward function is known. Solving an SDCOP optimally is then equivalent to solving a regular DCOP optimally, where the rewards of a constraint are the means of the respective reward functions specified by that constraint.

3.3 DPOP

Distributed Pseudo-tree Optimization Procedure (DPOP) [18] is a complete DCOP algorithm that can be viewed as a distributed version of the Bucket Elimination algorithm [3]. There are three phases in the operation of DPOP:

x_1	x_2	Rewards
0	0	max(5+ 5, 8+8) = 16
0	1	max(5+20, 8+3) = 25
1	0	max(20+ 5, 3+8) = 25
1	1	max(20+20, 3+3) = 40

(a)

x_1	Rewards
0	max(5+16, 8+25) = 33
1	max(20+25, 3+40) = 45

(b)

Table 1: UTIL Phase Computations in DPOP

(1) The first phase is the pseudo-tree generation phase, where DPOP calls existing distributed pseudo-tree construction algorithms like Distributed DFS [7] as a subroutine to construct its pseudo-tree.

(2) The second phase is the UTIL phase, where each agent, starting from the leafs of the pseudo-tree, computes the optimal sum of rewards in its subtree for each value combination of its ancestor agents. The agent does so by summing the rewards in the UTIL messages received from its child agents and projects out its own variable by optimizing over it. In our example problem, agent a_3 computes the optimal reward for each value combination of variables x_1 and x_2 as shown in Table 1(a) and sends the rewards to its parent agent a_2 in a UTIL message. Agent a_2 then computes the optimal reward for each value of variable x_1 as shown in Table 1(b) and sends the rewards to its parent agent a_2 in a UTIL message.

(3) The third phase is the VALUE phase, where each agent, starting from the root of the pseudo-tree, determines the optimal value for its variable. The root agent does so using the UTIL messages received in the second phase. In our example problem, agent a_1 determines from the UTIL message from agent a_2 that the value with the largest reward for its variable is 1 with a reward of 45. It then sends this information down to its child agent a_2 in a VALUE message. Upon receiving the VALUE message from its parent agent, agent a_2 determines that the value with the largest reward for its variable assuming that $x_1 = 1$ is 0 with a reward of 45. It then sends this information down to its child agent a_3 in a VALUE message. Finally, upon receiving the VALUE messages from its parent agent, agent a_3 determines that the value with the largest reward for its variable assuming that $x_1 = 1$ and $x_2 = 0$ is 0 with a reward of 25.

4. STOCHASTIC DOMINANCE

Although finding a solution that maximizes the expected reward in an SDCOP is a reasonable and intuitive objective, it fails to characterize "risk" very well. To illustrate this, Figure 3 shows two probability distribution functions, where the expected reward of the unimodal distribution is slightly smaller than that of the bimodal distribution but the variance is also smaller. Hence if the problem domain requires a decision maker to be risk-averse, then one might prefer the unimodal distribution over the bimodal one. In general, it is well-known (particularly in financial domains) that maximizing utility without considering risk does not yield good solutions in risky environments.

To incorporate the notion of risk, we propose a new (stricter) goal for SDCOPs, namely, our aim is to find a complete solution with the most stochastically dominating [6] overall reward function. Intuitively, a stochastically dominating function is less risky than any dominated function. In this paper, we will use the second-order stochastic dominance criteria [13] as the dominance criteria.

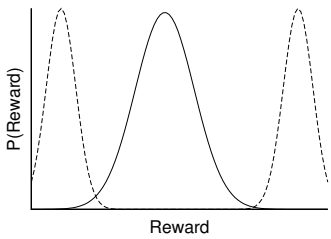


Figure 3: Unimodal and Bimodal Distributions

Let $f_{1+2+\ldots+m}(X)$ denote the overall reward function for a complete solution X, which is the convolution of the individual reward functions $f_i(x_{i_1}, x_{i_2} \mid x_{i_j} \in X)$ over all $i \in [0, m]$. And let $F_{X_k}(t) = \int_{-\infty}^{t} f_{1+2+\ldots+m}(X_k)(t)\, dt$ denote the corresponding cumulative distribution function. We say that a solution X_1 *dominates* (written $\prec$) another solution X_2 iff:

$$\int_{-\infty}^{x} F_{X_1}(t) - F_{X_2}(t)\, dt \leq 0 \tag{2}$$

for all values of x.

Our goal is hence to find a complete solution X^* with the most stochastically dominating overall reward function, i.e.:

$$f_{1+2+\ldots+m}(X^*) \not\prec f_{1+2+\ldots+m}(X) \tag{3}$$

Notice however that there may exist functions that do not dominate each other. As such, there can be a number of *pareto-optimal* solutions, where a pareto-optimal solution is a solution that is not dominated by any other solutions in the set.

Unfortunately, the agents in SDCOPs cannot compare the dominance of overall reward functions without transmitting the individual functions to a centralized agent. Therefore, we compute the dominance of the individual reward functions and find a complete solution X^* such that

$$\forall i : f_i(x^*_{i_1}, x^*_{i_2} \mid x^*_{i_j} \in X^*) \not\prec f_i(x_{i_1}, x_{i_2} \mid x_{i_j} \in X) \tag{4}$$

which implies Equation (3) according to Theorem 1, which we will prove in Section 5.4.

Moreover, it is guaranteed that for any arbitrary risk function, the optimal solution (i.e. the solution that maximizes the expected reward for that risk function) is a pareto-optimal solution [13]. Thus, finding the pareto set is useful in applications where the risk-averse function is not known *a priori* and users want to hedge against all possible risk functions. And once the risk function is known, we can find the optimal solution by evaluating only the pareto-optimal solutions (rather than all possible solutions).

5. SD-DPOP

We now introduce Stochastic Dominance DPOP (SD-DPOP), an extension of DPOP, to solve SDCOPs. The objective of SD-DPOP is to find the set of pareto-optimal solutions (when the risk function is not known *a priori*) and

x_1	x_2	Reward Functions
0	0	dom($\mathcal{N}$(5+ 5, 1+1), $\mathcal{N}$(8+8, 1+1)) = $\mathcal{N}$(16, 2)
0	1	dom($\mathcal{N}$(5+20, 1+1), $\mathcal{N}$(8+3, 1+1)) = $\mathcal{N}$(25, 2)
1	0	dom($\mathcal{N}$(20+ 5, 1+1), $\mathcal{N}$(3+8, 1+1)) = $\mathcal{N}$(25, 2)
1	1	dom($\mathcal{N}$(20+20, 1+1), $\mathcal{N}$(3+3, 1+1)) = $\mathcal{N}$(40, 2)

(a)

x_1	Reward Functions
0	dom($\mathcal{N}$(5+16, 1+2), $\mathcal{N}$(8+25, 1+2)) = $\mathcal{N}$(33, 3)
1	dom($\mathcal{N}$(20+25, 1+2), $\mathcal{N}$(3+40, 1+2)) = $\mathcal{N}$(45, 3)

(b)

Table 2: UTIL Phase Computations in SD-DPOP

Algorithm 1: SD-DPOP

```
   /* Phase 1:  Pseudo-tree Generation Phase     */
1  Generate pseudo-tree
   /* Phase 2:  UTIL Phase                       */
2  if x_i is a leaf node then
3  |  UTIL_{x_i} ← CalcUtils( );
4  |  Send UTIL message (UTIL_{x_i}) to parent
5  end
6  Activate UTILMessageHandler( )
   /* Phase 3:  VALUE Phase                      */
7  Activate VALUEMessageHandler( )
```

Procedure UTILMessageHandler($UTIL_{x_k}$)

```
8  Store UTIL_{x_k}
9  if received UTILmessage from every child then
10 |  if x_i is a root node then
11 |  |  v_i^* ← ChooseBestValue(NULL)
12 |  |  VALUE_{x_i} ← {(x_i, v_i^*)}
13 |  |  Send VALUE message (VALUE_{x_i}) to every child
14 |  else
15 |  |  UTIL_{x_i} ← CalcUtils( )
16 |  |  Send UTIL message (UTIL_{x_i}) to parent
17 |  end
18 end
```

Procedure VALUEMessageHandler($VALUE_{x_k}$)

```
19 VALUE_{x_i} ← VALUE_{x_k}
20 v_i^* ← ChooseBestValue(VALUE_{x_i})
21 VALUE_{x_i} ← VALUE_{x_i} ∪ {(x_i, v_i^*)}
22 Send VALUE message (VALUE_{x_i}) to every child
```

Function CalcUtils( )

```
23 UTIL_{x_i} ← Reward functions for all value combinations
     of x_i, its parent and pseudo-parents
24 UTIL_{x_i} ← Join(UTIL_{x_i}, UTIL_{x_c}) for all children x_c
25 UTIL_{x_i} ← Project(x_i, UTIL_{x_i})
26 Return UTIL_{x_i}
```

then find an optimal solution for the risk function (once the function is known). It finds the set of pareto-optimal solutions using stochastic dominance. It finds an optimal solution by evaluating the set of pareto-optimal solutions.

At a high level, SD-DPOP is similar to DPOP except that the agents convolve the reward functions instead of sum the rewards, choose values with the dominating convolved reward function instead of values with the maximum reward, and potentially find multiple pareto-optimal solutions instead of one reward maximizing solution.

5.1 High Level Description

Algorithm 1 shows the pseudo-code of a simplified version of SD-DPOP, where we assume that there is only one stochastically dominating solution to ease readability.[1] Like DPOP, there are also three phases in the operation of SD-DPOP. The first phase [Line 01] is identical to that of DPOP. The second phase [Lines 2-6] is similar to that of DPOP except for three differences:

(1) The function *Join*() [Line 24] joins all constraints involving x_i by setting the reward function for each value combination in the joined constraint to the convolution of the individual reward functions. In DPOP, the reward for each value combination is the sum of the individual rewards. In our example problem, agent a_3 convolves $g_{13} = f_{13}(x_1 = 0, x_3 = 0)$ and $g_{23} = f_{23}(x_2 = 0, x_3 = 0)$ to determine the reward function when $x_1 = 0$, $x_2 = 0$ and $x_3 = 0$:

$$\begin{aligned}\int_{-\infty}^{\infty} g_{13}(s)g_{23}(t-s)\,ds &= (g_{13} * g_{23})(t)\\ &= \mathcal{N}(5,1) * \mathcal{N}(5,1)\\ &= \mathcal{N}(5+5, 1+1)\\ &= \mathcal{N}(10,2)\end{aligned}$$

(2) The function *Project*() [Line 25] projects the joined constraint down on x_i by setting the reward function for each value combination in the projected constraint to the stochastically dominating reward function between all corresponding functions in the pre-projected constraints. In DPOP, the reward for each value combination is maximum over all corresponding rewards in the pre-projected constraints. In our example problem, agent a_3 sets the reward function for $x_1 = 0$, $x_2 = 0$ to the stochastically dominating reward functions between the functions for $x_1 = 0$, $x_2 = 0, x_3 = 0$ (which is $\mathcal{N}(5+5, 1+1) = \mathcal{N}(10, 2)$) and $x_1 = 0$, $x_2 = 0, x_3 = 1$ (which is $\mathcal{N}(8+8, 1+1) = \mathcal{N}(16, 2)$). Tables 2(a) and 2(b) show the Project() computations of agent a_3 and a_2, respectively, where the function *dom*() returns the stochastically dominating solution.

(3) The agents send the projected convolved reward functions instead of the maximum summed rewards to their parent agents in UTIL messages.

Thus, by the end of the second phase, the root agent knows

[1]One can easily extend this simplified version to find multiple pareto optimal solutions by having each agent send in its UTIL message a vector of reward functions in its pareto set, and (b) send in its VALUE message a vector of values corresponding to its value for each reward function in its pareto set.

the overall reward function of each pareto-optimal solution. Once the risk function is known, the root agent evaluates the overall reward function of each pareto-optimal solution to find an optimal solution for the given risk function. It then starts the third phase.

The third phase [Line 7] is similar to that of DPOP except that the function *ChooseBestValue()* [Lines 11 and 20] returns the value of the optimal solution. In DPOP, the function returns the value with the maximum reward. In our example problem, there is only one pareto-optimal solution and, thus, that solution is also the optimal solution for any given risk function. Agent a_1 thus chooses its optimal value 1 for its variable and sends this information down to its child agent a_2 in a VALUE message.[2] Upon receiving the VALUE message from its parent agent, agent a_2 determines that the optimal value for its variable assuming that $x_1 = 1$ is 0. Finally, upon receiving the VALUE messages from its parent agent, agent a_3 determines that the optimal value for its variable assuming that $x_1 = 1$ and $x_2 = 0$ is 0.

5.2 Implementation Details

We now describe how one implement the convolution and stochastic dominance determination of continuous and discrete reward functions:

Case 1: The reward functions are continuous. One can use the *conv*() and *int*() functions in Matlab to convolve two functions and compute integrals to check for stochastic dominance, respectively. However, if the reward functions are Gaussian functions, one can more efficiently convolve two functions – $\mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2) = \mathcal{N}(\mu_1, \sigma_1^2) * \mathcal{N}(\mu_2, \sigma_2^2)$ – and check the stochastic dominance of two functions – $\mathcal{N}(\mu_1, \sigma_1^2) \succ \mathcal{N}(\mu_2, \sigma_2^2)$ if $\mu_1 \geq \mu_2$ and $\sigma_1 \leq \sigma_2$ with at least either one being a strict inequality. Lastly, instead of checking all pairs of functions for stochastic dominance, one can also optimize this check by using the property that if $f_i \succ f_j$ and $f_j \succ f_k$, then $f_i \succ f_k$.

Case 2: The reward functions are discrete and are represented by samples in discretized bins. To convolve two reward functions, one can create a new sample whose value is the sum of a pair of samples, one from each reward function, for all possible pairs of samples. The new samples represent the convolved reward function. However, such a method does not scale up well with the number of samples since the number of operations is quadratic in the number of samples. We thus describe an optimized method whose number of operations is quadratic in the number of bins. Algorithm 2 shows the pseudocode, where function *Sample*$(k, f_i(j))$ returns the k-th sample in the j-th bin of function f_i and $n^j_{f_i}$ is the number of samples in that bin. We now associate a probability $P(x)$ with each sample x, which we use to compute the probability $P(f_i(j))$ and mean $\mu(f_i(j))$ of each bin j of each function f_i [Lines 1-4], which we use to create new samples [Lines 5-11]. One can further optimize the algorithm by ignoring empty bins and merging neighboring bins if the probability of one of the bin is less than a threshold.

[2]In the more complicated case where there might be multiple pareto-optimal solutions, each agent stores all the reward functions sent by its child agents and sends back the reward function corresponding to the optimal solution down to its child agents in VALUE messages. To reduce memory and message complexity, the agents can store and send hash functions of the reward functions instead of the actual reward functions.

Algorithm 2: Convolve(f_1, f_2)

```
1  foreach function f_i and bin j do
2      P(f_i(j)) ← Σ_{k=1}^{n^j_{f_i}} P(Sample(k, f_i(j))
3      μ(f_i(j)) ← (Σ_{k=1}^{n^j_{f_i}} P(Sample(k, f_i(j)) · Sample(k, f_i(j))) / P(f_i(j))
4  end
5  foreach bin i of function f_1 do
6      foreach bin j of function f_2 do
7          Create a new sample x ← μ(f_1(i)) + μ(f_2(j))
8          P(x) ← P(f_1(i)) · P(f_2(j))
9          Place x in the appropriate bin of the convolved
           function
10     end
11 end
```

To determine the stochastic dominance of two reward functions f_1 and f_2, one first computes the cumulative distribution function $F_i(t) = \sum_{k=1}^{t} f_i(k)$ of each reward function f_i. One then checks whether the following condition hold for all values of x: $\sum_{t \leq x} F_1(t) - F_2(t) \geq 0$. If so, then the reward function f_1 stochastically dominates f_2.

5.3 Complexity Analysis

In DPOP, VALUE messages contain only the value of the sending agent and UTIL messages contain a reward value for each combination of values of the parent and pseudo-parent of the sending agent. Thus, its message complexity is $O(maxDom^w)$, where $maxDom = \arg\max_i |d_i|$ and w is the induced width of the pseudo-tree.

In SD-DPOP, VALUE messages contain each pareto-optimal value of the sending agent and UTIL messages contain a representation of the reward function for each pareto-optimal solution and each combination of values of the parent and pseudo-parent of the sending agent. Thus, if the reward functions are continuous and one can represent the function analytically with a constant number of parameters, such as the Gaussian function, which can be represented just by the mean and variance, then the message complexity is $O(p \cdot maxDom^w)$, where p is the size of the pareto set. If the reward functions are discrete and are represented by samples in discretized bins, then the message complexity is $O(b \cdot p \cdot maxDom^w)$, where b is the largest number of bins used to represent one reward function.

Therefore, like DPOP, SD-DPOP also suffers from an exponential memory requirement. However, one can likely extend SD-DPOP to make it memory-bounded, similar to how researchers have bounded the memory of DPOP in extensions like MB-DPOP [19] and PC-DPOP [20].

5.4 Correctness and Completeness

The completeness of SD-DPOP follow quite trivially from that of DPOP since the main differences are that agents now convolve reward functions instead of sum up reward values and it chooses stochastically dominating reward functions instead of the maximal reward value.

We now prove the correctness of SD-DPOP. Let f_{i+j} denote the convolution of reward functions f_i and f_j and F_i denote the cumulative distribution function of f_i.

LEMMA 1. *For any two arbitrary reward functions f_i and f_j, $F_{i+j}(t) = \int_{-\infty}^{\infty} F_j(t-x)\, f_i(x)\, dx$*

PROOF: Let f_i and f_j be two arbitrary reward functions.

$$\begin{aligned} F_{i+j}(t) &\stackrel{\text{def}}{=} \int_{-\infty}^{t} f_{i+j}(y)\, dy \\ &= \int_{-\infty}^{t}\int_{-\infty}^{\infty} f_j(y-x)\, f_i(x)\, dx\, dy \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{t} f_j(y-x)\, f_i(x)\, dy\, dx \\ &= \int_{-\infty}^{\infty}\int_{-\infty}^{t} f_j(y-x)\, dy\, f_i(x)\, dx \\ &= \int_{-\infty}^{\infty} F_j(t-x)\, f_i(x)\, dx \end{aligned}$$ ■

LEMMA 2. *For any three arbitrary reward functions f_i, f_j and f_k, if $f_i \succ f_j$, then $f_{i+k} \succ f_{j+k}$.*

PROOF: Let f_i, f_j and f_k be three arbitrary reward functions where $f_i \succ f_j$.

$$\begin{aligned} &\int_{-\infty}^{z} F_{i+k}(t) - F_{j+k}(t)\, dt \\ &\quad= \int_{-\infty}^{z}\int_{-\infty}^{\infty} \big(F_i(t-x) - F_j(t-x)\big) f_k(x)\, dx\, dt \\ &\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(Lemma 1)} \\ &\quad= \int_{-\infty}^{\infty} f_k(x) \int_{-\infty}^{z} F_i(t-x) - F_j(t-x)\, dt\, dx \\ &\quad= \int_{-\infty}^{\infty} f_k(x) \int_{-\infty}^{z-x} F_i(y) - F_j(y)\, dy\, dx \end{aligned}$$

We know that (1) $\int_{-\infty}^{k} F_i(y) - F_j(y)\, dy \leq 0$ for all values of k according to Equation 2 since $f_i \succ f_j$ and (2) $\int_{-\infty}^{\infty} f_k(x)\, dx \geq 0$ since it is an integral of a probability distribution function. Combining both inequalities, we get $\int_{-\infty}^{\infty} f_k(x) \int_{-\infty}^{z-x} F_i(y) - F_j(y)\, dy\, dx \leq 0$, which implies that $f_{i+k} \succ f_{j+k}$ according to Equation 2. ■

THEOREM 1. *For any $2m$ arbitrary reward functions $f_{i_1}, \ldots, f_{i_m}, f_{j_1}, \ldots, f_{j_m}$, if $f_{i_k} \succ f_{j_k}$ for all values of k, then $f_{i_1+\ldots+i_m} \succ f_{j_1+\ldots+j_m}$.*

PROOF: Let $f_{i_1}, \ldots, f_{i_m}, f_{j_1}, \ldots, f_{j_m}$ be arbitrary reward functions where $f_{i_k} \succ f_{j_k}$ for all values of k.

$$\begin{aligned} f_{i_1+\ldots+i_m} &\succ f_{j_1+i_2+i_3+\ldots+i_m} && (f_{i_1} \succ f_{j_1} \text{ and Lemma 2}) \\ &\succ f_{j_1+j_2+i_3+\ldots+i_m} && (f_{i_2} \succ f_{j_2} \text{ and Lemma 2}) \\ &\succ \ldots \\ &\succ f_{j_1+j_2+j_3+\ldots+j_m} && (f_{i_m} \succ f_{j_m} \text{ and Lemma 2}) \end{aligned}$$

Thus, $f_{i_1+\ldots+i_m} \succ f_{j_1+\ldots+j_m}$. ■

Therefore, since convolving stochastically dominating individual reward functions result in a stochastically dominating overall reward function according to Theorem 1, SD-DPOP is correct.

6. RELATED WORK

Motivated by risk-sensitive applications, other researchers have also independently investigated the use of dominance to solve SDCOPs. For example, Stranders *et al.* defined dominance with respect to a given risk function, where a random variable X dominates a random variables Y for a given function U iff $\mathbb{E}[U(X+Z)] \geq \mathbb{E}[U(Y+Z)]$ for all possible random variables Z with strict inequality for at least one Z [22]. They then derived sufficient and necessary conditions for such dominance to hold for one example risk function, proposed U-GDL, an extension of the GDL algorithm that uses the new conditions, and experimentally demonstrated the benefits of their approach. Our work is different from theirs in that SD-DPOP uses a stochastic dominance criteria, which is applicable for the class of concave risk functions.

Another piece of related work is the Multi-Objective Max-Sum (MOMS) algorithm by Delle Fave *et al.*, which is an algorithm that finds pareto-optimal solutions for a set of objective functions [4]. Our work is different from theirs in that MOMS needs to know the set of risk functions (set of objectives) prior to finding the pareto-optimal solutions while SD-DPOP does not. SD-DPOP actually finds the set of pareto-optimal solutions for all possible risk functions, which is an infinite set. However, MOMS can find pareto-optimal solutions for arbitrary utility functions, while SD-DPOP can only do so for concave risk functions. However, we expect all these extensions to easily apply to all GDL-based DCOP algorithms including DPOP, Max-Sum [5] and Action-GDL [25].

7. EXPERIMENTAL EVALUATION

We now compare the stochastically dominating solutions found by the SD-DPOP algorithm to the solutions that maximize the expected reward found by DPOP. We run our experiments on a MacBook Pro with a 2.7GHz Intel Core i7 and 8GB memory. We use the same sensor network problem that we used as our motivating domain, where we arranged the sensors in a grid and each sensor can move in the four cardinal directions or stay stationary. Thus, each sensor has 5 possible values. Additionally, each sensor can be constrained with any one of its four neighboring sensors. We varied the size of the problem by increasing the number of sensors in the grid and the number of constraints. Each constraint between two sensors consists of 25 reward functions corresponding to the 25 pairs of possible values of the two sensors. Each reward function is a Gaussian function whose mean and variance are randomly generated between the ranges of 80 to 100 and 0 to 80, respectively.

We use the following risk functions to evaluate the utilities of the solutions found:

$$g_1(x, \alpha, th) = \begin{cases} \alpha\, x - (\alpha - 1)\, th & x \leq th \\ th & otherwise \end{cases}$$

$$g_2(x, \alpha, th) = -\exp(-\alpha\,(x - th)) + th$$

Both functions are commonly used risk functions in the literature [13, 21]. In both functions, the parameter α represents the level of risk aversion. The larger the value of α, the higher the level of risk aversion. The parameter th represents the threshold after which one obtains close to no utility with increasing x.

We run experiments for both continuous and discretized

(a) SD-DPOP with Stochastic Dominating Solutions

no. of agents	no. of constraints	runtime (ms)	msg size (kB)	$g_1(x, 50, th)$			$g_1(x, 0, th)$			$g_2(x, 0.05, th)$			$g_2(x, 0.025, th)$		
				μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R
9	12	539	368	1391	92	1160	1412	103	1292	1369	84	1194	1390	91	1199
16	24	2588	7240	2613	142	2176	2611	141	2395	2542	115	-3261	2571	123	2397
25	37	81821	65963	4055	177	3619	4058	179	3698	3971	151	-1596	4004	159	3698

(b) DPOP with Expected Reward Maximizing Solutions

no. of agents	no. of constraints	runtime (ms)	msg size (kB)	$g_1(x, 50, th)$			$g_1(x, 0, th)$			$g_2(x, 0.05, th)$			$g_2(x, 0.025, th)$		
				μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R
9	12	100	75	1455	167	949	1455	167	1283	1455	167	-2.13E8	1455	167	11260
16	24	172	139	2689	225	1840	2689	225	2389	2689	225	-1.66E14	2689	225	-7336
25	37	252	215	4162	278	3397	4162	278	3694	4162	278	-1.34E16	4162	278	-775615

Table 3: Experimental Results for Continuous Reward Functions

(a) SD-DPOP with Stochastic Dominating Solutions for 9 Agents and 9 Constraints

no. of samples	runtime (ms)	msg size (kB)	error in μ	error in σ	$g_1(x, 50, th)$			$g_1(x, 0, th)$			$g_2(x, 0.05, th)$			$g_2(x, 0.025, th)$		
					μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R
10	3203	2417	91	185	971	86	373	971	86	889	962	80	-1472	965	81	897
100	7775	3032	25	44	971	73	544	972	74	893	959	67	849	965	69	898
1000	12786	2909	30	41	972	74	546	972	74	893	958	66	847	968	71	898
10000	15802	2803	31	43	972	74	546	972	75	893	957	66	854	966	70	898

(b) SD-DPOP with Stochastic Dominating Solutions for 16 Agents and 16 Constraints

no. of samples	runtime (ms)	msg size (kB)	error in μ	error in σ	$g_1(x, 50, th)$			$g_1(x, 0, th)$			$g_2(x, 0.05, th)$			$g_2(x, 0.025, th)$		
					μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R	μ	σ	μ_R
10	35845	9458	340	258	1751	132	1143	1751	132	1591	1741	128	-2.76E7	1741	127	1562
100	58125	9792	65	85	1750	105	1399	1750	105	1596	1721	93	1092	1736	97	1599
1000	69050	9088	80	59	1751	106	1402	1750	105	1596	1719	91	1057	1734	97	1599
10000	79235	8869	79	58	1750	104	1403	1750	105	1596	1717	90	1178	1734	96	159

Table 4: Experimental Results for Discretized Reward Functions

reward functions. Tables 3 and 4 show our experimental results, where we average over 50 problem instances for each configuration. For each problem instance, we run SD-DPOP to find the pareto set, and for each risk function, we evaluate the risk of a pareto-optimal solution by taking 10,000 samples of the overall reward function for that solution and evaluating those samples using the risk function. We then return the solution with the largest mean as the optimal solution. The root agent performs this sampling process since it knows the overall reward function for each pareto-optimal solution. After it determines the optimal solution, it propagates that information down the pseudo-tree to the other agents.

For each problem configuration, we use four risk functions: $g_1(x, 50, th)$, $g_1(x, 0, th)$, $g_2(x, 0.05, th)$ and $g_2(x, 0.025, th)$, where we set th to 100 times the number of constraints in the problem. For each risk function, we report the mean (μ) and standard deviation (σ) of the overall reward function of the optimal solution as well as the mean (μ_R) of the evaluation of overall reward function using the risk function. We also report the error in the mean and standard deviation (compared against the true mean and standard deviation) of the overall reward function of the optimal solution for experiments with discretized reward functions. We make the following observations:

- The runtime and message size of DPOP and SD-DPOP increases with the number of agents and constraints, which is to be expected. The runtime and message size of SD-DPOP is larger than that of DPOP. The runtime is larger because agents in SD-DPOP find multiple pareto-optimal solutions while agents in DPOP find only one solution. The message size is larger because agents in SD-DPOP sends more information in each message, as described in Section 5.3.
- The means (μ) of the expected reward maximizing solutions are larger than the means of the stochastic dominating solutions, which is to be expected. However, the means (μ_R) of the evaluations of the expected reward maximizing solutions using the risk functions are smaller than those of the stochastic dominating solutions, which implies that the solutions found by SD-DPOP is less risky. The difference in the means increases as α increases. Thus, the higher the level of risk aversion, the more important it is to find stochastic dominating solutions. An interesting future direction of research would be to compute a theoretical bound between μ and μ_R. However, this bound would be useful for risk functions with small values of α only. The reason is that if α is large or, equivalently, the risk function has an almost vertical slope, then μ is likely to be finite while μ_R is likely to be negative infinity.
- The runtime of SD-DPOP increases with the number of samples, which is to be expected. The message size increases as we increase the number of samples from 10 to 100. The reason is that the number of bins needed by the samples increases. However, the message size

does not increase further, and can sometimes decrease instead, as we increase the number of samples to 1000 or 10000. The reason is that as the number of samples increases significantly, the samples all converge near the mean of the distribution. As a result, the probability of bins at the tail ends become small. Since we optimize the number of bins by merging neighboring bins if the probability of one of the bin is less than a threshold of 0.001, these bins at each of the tail ends are merged into a single large bin.

- The errors in the mean and standard deviation decrease as we increase the number of samples from 10 to 100. The reason is that 10 samples is too small to accurately represent the reward function. However, the errors do not decrease further, and actually increase instead, as we increase the number of samples to 1000 or 10000. The reason is that 100 samples is sufficient to accurately represent the reward function and the increase in error is likely due to sampling approximations.

8. CONCLUSIONS

The Stochastic DCOP (SDCOP) model is useful in modeling multi-agent coordination problems where the constraint rewards are sampled from known reward functions. The goal of SDCOPs is to find a solution that maximizes the expected reward in such problems. However, these solutions might be very risky and hence unacceptable in risk-sensitive applications. In this paper, we make three contributions: (1) we propose a stricter objective for SDCOPs, namely to find a solution with the most stochastically dominating reward function; (2) we introduce SD-DPOP, an extension of DPOP that finds such solutions; and (3) we show that stochastically dominating solutions can indeed be less risky than expected reward maximizing solutions. For future work, we would like to bound the memory used by SD-DPOP in the same way that researchers have done for MB-DPOP and PC-DPOP. We believe that this is important since the exponential requirement on memory prohibits the use of this algorithm in large scale problems.

9. ACKNOWLEDGMENT

This research is supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office. We would also like to thank our anonymous reviewers for their comments and suggestions, especially for pointing us to the work on the U-GDL and MOMS algorithms.

10. REFERENCES

[1] J. Atlas and K. Decker. Coordination for uncertain outcomes using distributed neighbor exchange. In *Proc. of AAMAS*, pages 1047–1054, 2010.

[2] D. Burke and K. Brown. Efficiently handling complex local problems in distributed constraint optimisation. In *Proc. of ECAI*, pages 701–702, 2006.

[3] R. Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85, 1999.

[4] F. Delle Fave, R. Stranders, A. Rogers, and N. Jennings. Bounded decentralised coordination over multiple objectives. In *Proc. of AAMAS*, pages 371–378, 2011.

[5] A. Farinelli, A. Rogers, A. Petcu, and N. Jennings. Decentralised coordination of low-power embedded devices using the Max-Sum algorithm. In *Proc. of AAMAS*, pages 639–646, 2008.

[6] S. Graves and J. Ringuest. Probabilistic dominance criteria for comparing uncertain alternatives: A tutorial. *Omega*, 37(2):346–357, 2009.

[7] Y. Hamadi, C. Bessière, and J. Quinqueton. Distributed intelligent backtracking. In *Proc. of ECAI*, pages 219–223, 1998.

[8] M. Jain, M. Taylor, M. Tambe, and M. Yokoo. DCOPs meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks. In *Proc. of IJCAI*, pages 181–186, 2009.

[9] S. Kozono. Received signal-level characteristics in a wide-band mobile radio channel. *IEEE Transactions on Vehicular Technology*, 43(3):480–486, 1994.

[10] A. Kumar, B. Faltings, and A. Petcu. Distributed constraint optimization with structured resource constraints. In *Proc. of AAMAS*, pages 923–930, 2009.

[11] R. Lass, J. Kopena, E. Sultanik, D. Nguyen, C. Dugan, P. Modi, and W. Regli. Coordination of first responders under communication and resource constraints (Short Paper). In *Proc. of AAMAS*, pages 1409–1413, 2008.

[12] T. Léauté and B. Faltings. $\mathbb{E}$[DPOP]: Distributed constraint optimization under stochastic uncertainty using collaborative sampling. In *Proc. of DCR*, pages 87–101, 2009.

[13] H. Levy. *Stochastic Dominance Investment Decision Making under Uncertainty Studies in Risk and Uncertainty*. Springer, 1998.

[14] V. Lisỳ, R. Zivan, K. Sycara, and M. Péchoucek. Deception in networks of mobile sensing agents. In *Proc. of AAMAS*, pages 1031–1038, 2010.

[15] R. Maheswaran, M. Tambe, E. Bowring, J. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling. In *Proc. of AAMAS*, pages 310–317, 2004.

[16] M. McClure, D. Corbett, and D. Gage. The DARPA LANdroids program. *SPIE*, 7332:73320A, 2009.

[17] P. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

[18] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proc. of IJCAI*, pages 1413–1420, 2005.

[19] A. Petcu and B. Faltings. MB-DPOP: A new memory-bounded algorithm for distributed optimization. In *Proc. of IJCAI*, pages 1452–1457, 2007.

[20] A. Petcu, B. Faltings, and R. Mailler. PC-DPOP: A new partial centralization algorithm for distributed optimization. In *Proc. of IJCAI*, pages 167–172, 2007.

[21] J. Pratt. Risk aversion in the small and in the large. *Econometrica*, 32(1–2):122–136, 1964.

[22] R. Stranders, F. Delle Fave, A. Rogers, and N. Jennings. U-GDL: A decentralised algorithm on DCOPs with uncertainty. Technical report, Department of Electronics and Computer Science, University of Southampton, 2011.

[23] M. Taylor, M. Jain, Y. Jin, M. Yokoo, and M. Tambe. When should there be a "me" in "team"?: Distributed multi-agent optimization under uncertainty. In *Proc. of AAMAS*, pages 109–116, 2010.

[24] S. Ueda, A. Iwasaki, and M. Yokoo. Coalition structure generation based on distributed constraint optimization. In *Proc. of AAAI*, pages 197–203, 2010.

[25] M. Vinyals, J. Rodríguez-Aguilar, and J. Cerquides. Constructing a unifying theory of dynamic programming DCOP algorithms via the Generalized Distributive Law. *Autonomous Agents and Multi-Agent Systems*, 2010.

Max/Min-sum Distributed Constraint Optimization through Value Propagation on an Alternating DAG

Roie Zivan and Hilla Peled
Industrial Engineering and Management department,
Ben Gurion University,
Beer-Sheva, Israel
{zivanr,hillapel}@bgu.ac.il

ABSTRACT

Distributed Constraint Optimization Problems (DCOPs) are NP-hard and therefore the number of studies that consider incomplete algorithms for solving them is growing. Specifically, the Max-sum algorithm has drawn attention in recent years and has been applied to a number of realistic applications. Unfortunately, in many cases Max-sum does not produce high quality solutions. More specifically, when problems include cycles of various sizes in the factor graph upon which Max-sum performs, the algorithm does not converge and the states that it visits are of low quality.

In this paper we advance the research on incomplete algorithms for DCOPs by: (1) Proposing a version of the Max-sum algorithm that operates on an alternating directed acyclic graph (Max-sum_AD), which guarantees convergence in linear time. (2) Identifying major weaknesses of Max-sum and Max-sum_AD that cause inconsistent costs/utilities to be propagated and affect the assignment selection. (3) Solving the identified problems by introducing value propagation to Max-sum_AD. Our empirical study reveals a large improvement in the quality of the solutions produced by Max-sum_AD with value propagation (VP), when solving problems which include cycles, compared with the solutions produced by the standard Max-sum algorithm, Bounded Max-sum and Max-sum_AD with no value propagation.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

DCOP, Incomplete Algorithms, GDL

1. INTRODUCTION

The Distributed Constraint Optimization Problem (DCOP) is a general model for distributed problem solving that has a wide range of applications in Multi-Agent Systems and has generated significant interest from researchers [10, 12, 21, 15, 16].

A number of studies on DCOPs presented complete algorithms [10, 12, 5]. However, since DCOPs are NP-hard, there is a growing interest in the last few years in incomplete DCOP algorithms [9, 21, 17]. Although incomplete algorithms do not guarantee that the obtained solution is optimal, they are applicable for large problems and compatible with real time applications.

Local search algorithms for DCOPs are incomplete algorithms whose general structure is synchronous. In each step of the algorithm an agent sends its assignment to all its neighbors in the constraint network and receives the assignment of all its neighbors. They differ in the method agents use to decide whether to replace their current value assignments to their variables, e.g., in the max gain messages algorithm (MGM) [9]; the agent that can improve its state the most in its neighborhood replaces its assignment. A stochastic decision whether to replace an assignment is made by agents in the distributed stochastic algorithm (DSA) [21].

An incomplete algorithm that does not follow the standard structure of distributed local search algorithms and has drawn much attention recently is the Max-sum algorithm [4]. Max-sum is an incomplete *GDL* algorithm [1]. In contrast to standard local search algorithms, agents in Max-sum do not propagate assignments but rather calculate utilities (or costs) for each possible value assignment of their neighboring agents' variables. The general structure of the algorithm is exploitive, i.e., the agents attempt to compute the best costs/utilities for possible value assignments according to their own problem data and recent information they received via messages from their neighbors.

The growing interest in the Max-sum algorithm in recent years included its use for solving DCOPs representing various multi-agent applications, e.g., sensor systems [17, 15] and task allocation for rescue teams in disaster areas [13]. In addition, a method for approximating the distance of the solution found by Max-sum from the optimal solution for a given problem was proposed [14]. This version required the elimination of some of the problem's constraints in order to reduce the DCOP to a tree structured problem that can be solved in polynomial time. Then, the sum of the worst costs for all eliminated constraints serves as the bound on the approximation of the optimal solution.

Previous studies have revealed that Max-sum does not always converge to a solution [4]. In fact, in some of the cases where it does not converge, it traverses states with low quality and thus, at the end of the run a poor quality solution is reported. This pathology apparently occurs when the constraint graph of the problem includes cycles of various sizes [4]. Unfortunately, many DCOPs that were investigated in previous studies are dense and indeed include such cycles (e.g., [10, 5]). Our experimental study revealed that on random problems of different density parameters and problem sizes, and on problems with structured constraints (graph coloring), Max-sum does not converge.

In this paper we contribute to the development of incomplete

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

algorithms for solving DCOPs by:

1. Proposing a new version of the Max-sum algorithm that uses an alternating directed acyclic graph (DAG). The proposed algorithm (Max-sum_AD) avoids cycles by performing iterations of the algorithm in which messages are sent according to a predefined order. The order divides the set of neighbors for each agent to two disjoint subsets, one of agents that come before it in the order in which it receives messages, and the other of neighbors that come after it in the order in which it sends messages (notice that the order is on the direction of messages and not on the agents' actions).

   In order not to ignore constraints of the DCOP, after a number of iterations in which all agents perform concurrently, which guarantees the convergence of the algorithm, the order from which the direction of the DAG is derived is reversed. Then, the algorithm is performed on the reversed DAG until it converges again. We prove that the maximal number of iterations in a single direction required for the algorithm to converge is equal to the longest path in the DAG, l (linear in the worst case). Thus, by performing l iterations in each direction we converge to a solution after considering all the constraints in the DCOP.

2. We identify major weaknesses of Max-sum and Max-sum_AD, which are their performance in the presence of ties and the possibility that best costs/utilities computed for the same variable consider different value assignments and thus are not valid. We demonstrate that the propagation of such inconsistent information results in a distorted selection of assignments by the agents.

3. We solve the problems of inconsistent cost propagation and tie breaking by using value propagation. After performing the algorithm in both directions and allowing it to converge for the second time after considering all the problem's constraints, we require that agents add to the messages they send the value assignment they have selected and that only constraints with these values are considered. Thus, we prevent the possibility that different agents consider costs/utilities that are based on conflicting value assignments. We note that value propagation has been used in previous studies for complete GDL algorithms [12, 18]) for breaking ties, including for the production of the optimal solution for the tree structure factor graph in Bounded Max-sum [14]. However, to best of our knowledge, ours is the first use of value propagation in GDL algorithms for solving DCOPs with cyclic factor graphs.

Our empirical study demonstrates the success of Max-sum_AD combined with value propagation in comparison with the standard Max-sum algorithm and with Bounded Max-sum when solving random DCOPs and graph coloring problems (problems that include cycles on which Max-sum fails to converge).

The rest of this paper is organized as follows: We present related work in Section 2. DCOPs are presented in Section 3. Section 4 presents the standard Max-sum algorithm. The Max-sum_AD algorithm is presented in Section 5. Section 6 identifies the need for value propagation (VP) and further describes how VP is combined with Max-sum_AD. Section 7 includes an evaluation of the proposed algorithm in comparison with Max-sum and Bounded Max-sum. In Section 8 we discuss how dynamic spanning trees can be generated as part of the Max-sum_AD algorithm which can be used to produce bounded approximation of the optimal solution. Our conclusions are presented in Section 9.

2. RELATED WORK

Many algorithms for solving DCOPs were proposed in the last decade. These can be divided into complete and incomplete algorithms. While some complete algorithms such as ADOPT, BnB-ADOPT [10, 20], and AFB [5] perform distributed search, GDL-based complete algorithms implement a dynamic programming approach [1, 12, 18]. The first to apply this approach to DCOP were Petcu and Faltings by proposing the DPOP algorithm [12]. DPOP performs dynamic programming on a pseudo-tree structure. Since pseudo trees may have limited branching in dense problems, recent studies investigated alternative structures (e.g., junction trees) in order to increase the parallelism in GDL based algorithms [3, 18].

The other set of incomplete algorithms for solving DCOPs includes search algorithms as well. As mentioned in the Introduction, in standard local search algorithms for DCOPs (e.g., DSA) agents exchange messages in synchronous iterations and the algorithms differ in the method agents use to decide when and how to replace their assignment.

In [9, 11], a different approach towards local search for solving DCOPs was proposed. In these studies, completely exploitive algorithms are used to converge to local optima solutions, which are guaranteed to be within a predefined distance from the global optimal solution. The approximation level is dependent on a parameter k, which defines the size of coalitions that agents can form. These k size coalitions transfer the problem data to a single agent, which performs a complete search procedure in order to find the best assignment for all agents within the k size coalition. As a result, the algorithm converges to a state that is *k optimal* (k-opt) [11], i.e., no better state exists if k agents or fewer change their assignments.

The production of k-opt solutions may require solving an exponential number of problems of size k. To overcome this shortcoming, recent studies have proposed alternatives for the selection of small local environments that would be solved optimally in order to produce quality guarantees on the overall solution. One alternative, t-distance, created environments dependent on the distance of nodes in the constraint network [6]. While this alternative reduced the number of problems that need to be solved it did not bound the size of the problems that are solved. The most recent approach included the generation of environments that were bounded both by distance and size [19]. Thus, the number of problems to solve is bounded by the number of agents and the problems to solve by the predefined size.

Aggregation of agents' constraints was also used in an attempt to cope with the in-convergence of Max-sum [4]. It included the union of groups of agents to clusters of adjacent agents represented by a single agent in the cluster. The constraints between the agents in the cluster were aggregated and held by the agent representing the cluster. Thus, it required that some constraints would be revealed in a preprocessing phase to agents that are not included in the constraints (the constraint between agents A_1 and A_2 is revealed to agent A_3). Furthermore, the amount of aggregated information was not limited and in dense problems could result in a single agent holding a large part of the problem's constraints (partial centralization). Another approach is to aggregate constraints and unite nodes in the constraint graph so that the resulting graph would be a tree [18]. However, the result of this rearrangement of the constraint graph is the need to perform exponential computation and transfer exponential communication that will result in a complete solution.

In this paper we focus on incomplete GDL algorithms that avoid partial centralization and clustering of agents, and attempt to solve the original DCOPs as do standard complete algorithms (e.g., ADOPT and DPOP), *one-opt* distributed local search algorithms (e.g., DSA

and MGM) and as the standard Max-sum algorithm does [14].

The alternating direction approach we implement in this paper is inspired by algorithms for solving asymmetric distributed constraints problems [2]. However, unlike in the case of asymmetric problems where the motivation for this approach was preserving privacy, in this paper the motivation is strictly algorithmic.

3. DISTRIBUTED CONSTRAINT OPTIMIZATION

To avoid confusion, and without loss of generality, in the rest of this paper we will assume all problems are minimization problems as presented in the early DCOP papers (e.g., [10]). Thus, we assume that all constraints define costs and not utilities. The GDL algorithm for minimization problems is actually a *Min-sum* GDL algorithm. However, we will continue to refer to it as Max-sum since this name is widely accepted. Our description of a DCOP is also consistent with the definitions in many DCOP studies, e.g., [10, 12, 5].

A $DCOP$ is a tuple $\langle \mathcal{A}, \mathcal{X}, \mathcal{D}, \mathcal{R} \rangle$. $\mathcal{A}$ is a finite set of agents $A_1, A_2, ..., A_n$. $\mathcal{X}$ is a finite set of variables X_1,X_2,...,X_m. Each variable is held by a single agent (an agent may hold more than one variable). $\mathcal{D}$ is a set of domains D_1, D_2,...,D_m. Each domain D_i contains the finite set of values that can be assigned to variable X_i. We denote an assignment of value $d \in D_i$ to X_i by an ordered pair $\langle X_i, d \rangle$. $\mathcal{R}$ is a set of relations (constraints). Each constraint $C \in \mathcal{R}$ defines a non-negative *cost* for every possible value combination of a set of variables, and is of the form $C : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \rightarrow \mathbb{R}^+ \cup \{0\}$. A *binary constraint* refers to exactly two variables and is of the form $C_{ij} : D_i \times D_j \rightarrow \mathbb{R}^+ \cup \{0\}$. A *binary DCOP* is a DCOP in which all constraints are binary. A *partial assignment* (PA) is a set of value assignments to variables, in which each variable appears at most once. *vars(PA)* is the set of all variables that appear in PA, $vars(PA) = \{X_i \mid \exists d \in D_i \wedge \langle X_i, d \rangle \in PA\}$. A constraint $C \in \mathcal{R}$ of the form $C : D_{i_1} \times D_{i_2} \times \ldots \times D_{i_k} \rightarrow \mathbb{R}^+ \cup \{0\}$ is *applicable* to PA if $X_{i_1}, X_{i_2}, \ldots, X_{i_k} \in vars(PA)$. The *cost of a partial assignment* PA is the sum of all applicable constraints to PA over the assignments in PA. A *complete assignment* is a partial assignment that includes all the variables ($vars(PA) = \mathcal{X}$). An *optimal solution* is a complete assignment with minimal cost.

For simplicity, all DCOPs considered in this paper are binary DCOPs in which each agent holds exactly one variable.

4. STANDARD MAX-SUM

The Max-Sum algorithm [4] is a GDL algorithm [1] that operates on a *factor graph* [7] that is a bipartite graph in which the nodes represent variables and constraints. [1] Each node representing a variable of the original DCOP is connected to all function-nodes that represent constraints that it is involved in. Similarly, a function-node is connected to all variable-nodes that represent variables in the original DCOP that are included in the constraint it represents. Agents in Max-sum perform the roles of different nodes in the factor graph. We will assume that each agent takes the role of the variable-nodes that represent its own variables and for each function-node, one of the agents whose variable is involved in the constraint it represents, performs its role. Variable-nodes and function-nodes are considered "agents" in Max-sum, i.e., they can send messages, read messages and perform computation.

Figure 1 demonstrates the transformation of a DCOP to a factor graph. On the top we have a DCOP with three agents, each holding a single variable. All variables are connected by binary constraints. On the bottom we have a factor graph. Each agent takes the role of the node representing its own variable and the role of one of the function-nodes representing a constraint it is involved in, e.g., in this factor graph agent A_1 takes the role of function-node f_1 which represents the constraint between its own variable x_1 and variable x_3 held by agent A_3.

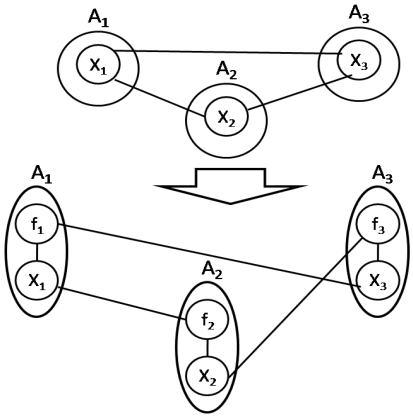


Figure 1: Transformation of a DCOP to a factor graph

Max-sum (node n)
1. $N_n \leftarrow$ all of n's neighboring nodes
2. **while** (no termination condition is met)
3. collect messages from N_n
4. **for each** $n' \in N_n$
5. **if** (n is a variable-node)
6. produce message $m_{n'}$ using messages from $N_n \setminus \{n'\}$
7. **if** (n is a function-node)
8. produce message $m_{n'}$ using constraint and messages from $N_n \setminus \{n'\}$
9. send $m_{n'}$ to n'

Figure 2: Standard Max-sum.

Figure 2 presents a sketch of the Max-sum algorithm.[2] The code for variable-nodes and function-nodes is similar apart from the computation of the content of messages to be sent. For variable-nodes only data received from neighbors is considered. In messages sent by function-nodes the content is produced considering data received from neighbors and the original constraint represented by the function-node.

It remains to describe the content of messages sent by the factor graph nodes. A message sent from a variable-node x to a function-node f at iteration i, includes for each of the values $d \in D_x$ the sum of costs for this value it received from all function neighbors apart from f in iteration $i-1$. Formally, for value $d \in D_x$ the message will include:

$$\sum_{f' \in F_x, f' \neq f} cost(f'.d) - \alpha$$

where F_x is the set of function-node neighbors of variable x and $cost(f'.d)$ is the cost for value d included in the message received from f' in iteration $i-1$. α is a constant that is reduced from all costs included in the message (i.e., for each $d \in D_x$) in order to prevent the costs carried by messages throughout the algorithm

[1] We preserve the terminology of [4] and call constraint representing nodes in the factor graph "function-nodes".

[2] In contrast to previous papers on Max-sum, we present it using a pseudo-code. This is following standard DCOP literature, e.g., [10, 12, 21]. Nevertheless, only the presentation is different. The algorithm itself is identical to the algorithm presented in [4, 14].

from growing arbitrarily. Selecting α to be the average on all costs included in the message is a reasonable choice for this purpose [4, 14]. Notice that as long as the amount reduced from all costs is identical, the algorithm is not affected by this reduction since only the differences between the costs for the different values matter.

A message sent from a function-node f to a variable-node x in iteration i includes for each possible value $d \in D_x$ the minimal cost of any combination of assignments to the variables involved in f apart from x and the assignment of value d to variable x. Formally, the message from f to x includes for each value $d \in D_x$:

$$min_{ass_{-x}} cost(\langle x, d\rangle, ass_{-x})$$

where ass_{-x} is a possible combination of assignments to variables involved in f not including x. The cost of an assignment $a = (\langle x, d\rangle, ass_{-x})$ is:

$$f(a) + \sum_{x' \in X_f, x' \neq x} cost(x'.d')$$

where $f(a)$ is the original cost in the constraint represented by f for the assignment a, X_f is the set of variable-node neighbors of function-node f, and $cost(x'.d')$ is the cost that was received in the message sent from variable-node x' in iteration $i-1$, for the value d' that is assigned to x' in a.

While the selection of value assignments to variables is not used to generate the messages in the Max-sum algorithm, in every iteration an agent can select its value assignment and the assignment selected by agents at the end of the run is the reported solution. Each variable-node selects the value assignment that received the lowest sum of costs included in the messages that were received most recently from its neighboring function-nodes. Formally, for variable x we select the value $\hat{d} \in D_x$ as follows:

$$\hat{d} = min_{d \in D_x} \sum_{f \in F_x} cost(f.d)$$

Notice that the same information used by the variable-node to select the content of the messages it sends is used for selecting its value assignment.

5. MAX-SUM ON AN ALTERNATING DAG (MAX-SUM_AD)

In order to guarantee the convergence of Max-sum we need to avoid the pathology described in [4], caused by cycles in the factor graph. To this end we select an order on all nodes in the factor graph. For example, we can order nodes according to the indices of agents performing their role in the algorithm. A node whose role is performed by agent A_i is ordered before a node whose role is performed by agent A_j if $i < j$. For variable and function nodes held by the same agent, we can determine (without loss of generality) that a variable-node is ordered before the function-nodes and break ties among function-nodes using their indices.

Once we define an order on all nodes in the factor graph, each agent (node) can divide its set of neighbors to two disjoint subsets, the subset of neighbors in the factor graph that come before it in the order, from whom it receives messages, and the subset of neighbors that are ordered after it, to whom it sends messages.

Next, we perform the algorithm for l iterations allowing nodes to send messages only to nodes which are "after" them according to this order (in the case of ordering by indices, send messages only to agents with larger indices than their own). Notice that all agents perform concurrently in each iteration of the algorithm, thus the order does not affect the agents' actions, only the direction of messages.

Max-sum_AD (node n)

1. $currebt\_order \leftarrow$ select an order on all nodes in the factor graph
2. $N_n \leftarrow$ all of n's neighboring nodes
3. **while** (no termination condition is met)
4. $\quad N_{prev\_n} \leftarrow \{\hat{n} \in N_n :$ $\hat{n}$ is before n in $current\_order\}$
5. $\quad N_{follow\_n} \leftarrow N_n \setminus N_{prev\_n}$
6. $\quad$ **for**(k iterations)
7. $\quad\quad$ collect messages from N_{prev_n}
8. $\quad\quad$ **for each** $n' \in N_{follow\_n}$
9. $\quad\quad\quad$ **if** (n is a variable-node)
10. $\quad\quad\quad\quad$ produce message $m_{n'}$ using messages from $N_n \setminus \{n'\}$
11. $\quad\quad\quad$ **if** (n is a function-node)
12. $\quad\quad\quad\quad$ produce message $m_{n'}$ using constraint and messages received from $N_n \setminus \{n'\}$
13. $\quad\quad\quad$ send $m_{n'}$ to n'
14. $\quad current\_order \leftarrow reverse(current\_order)$

Figure 3: Max-sum_AD.

After l iterations in the selected direction, the order is reversed and messages are sent for the next l iterations only in the opposite direction (i.e., to agents with lower indices). In each direction the Max-sum algorithm is performed as described in Section 4 with the exception of the restriction on the messages that are sent. Agents always consider the last message received from all their neighbors (regardless of the direction) when producing a new message. For example, when variable-node x produces a message it would send to function-node f, all of the most recent messages x received from its neighboring functions $f' \in F_x$, $f' \neq f$ are considered. Notice that the most recent message from a neighbor that is before x according to the current order was received following the previous iteration, while from a neighboring function-node that is after x according to the current order, the last message was received before the last alternation of directions.

The resulting algorithm Max-sum_AD has messages sent according to a directed acyclic graph (DAG), which is determined by the current order. Each time the order changes, we get a DAG on which messages on each edge of the graph are sent only in a single direction.

Figure 3 presents a sketch of the Max-sum_AD algorithm. It differs from standard Max-sum in the selection of directions and the disjoint sets of neighbors from whom the nodes receive messages and to whom they send messages (lines 1, 4, 5 and 14).
Next we prove the convergence of Max-sum_AD when performed in a single direction.

LEMMA 1. *Given o, an order on the nodes of the factor graph FG, for any node $n \in FG$, if l is the length of the longest path in FG according to o that reaches n, then after l iterations, the content of the messages n receives does not change as long as messages are sent according to o.*

Proof: We prove by a complete induction on the length of the longest path to node n according to o. A node n', which is first according to o, i.e., the length of the longest path that reaches it according to o is equal to zero, does not receive messages from any other node. We assume the correctness of the Lemma for any node n' that the longest path that reaches it is $l' < l$. We now check the correctness of the Lemma for node n, where the longest path reaching it is equal to l. The longest path to all of the neighbors of node n that are ordered before it according to o must be shorter than l. Thus, according to the assumption, after $l-1$ iterations of

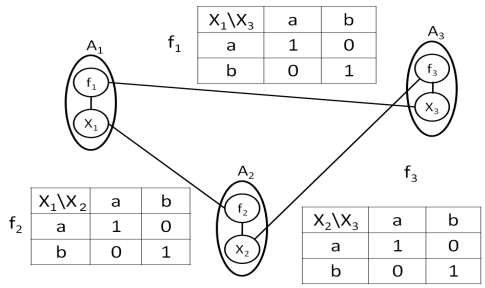


Figure 4: Example of the need to break ties

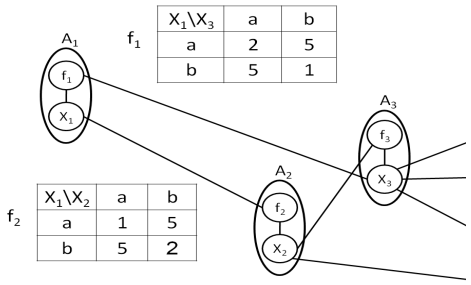


Figure 5: Example of the need for value propagation beyond ties

the algorithm the messages they receive will not change. Since the content of messages produced by agents in the Max-sum algorithm is dependent on the content of messages they received last, then from the $l'th$ iteration and on these neighbors of node n will be sending identical messages to n. Thus, the Lemma holds for node n as well. □.

The first immediate corollary from Lemma 1 is that after a number of iterations equal to the diameter of the factor graph FG, all the nodes in FG will continue to receive the same messages until o is reversed. Moreover, since agents use the last messages they have received in order to select their value assignments, the value assignments will not change either. Thus, the algorithm converges to a single complete assignment.

The decision to escape this fixed assignment by changing direction is an algorithmic decision. If this decision is made, the algorithm will converge again after a linear number of iterations in the reversed order, but not necessarily to a better solution. We demonstrate in our empirical study that one version of Max-sum_AD monotonically improves the states it converges to after each direction change and that another does not.

Notice, that after the first alternation of direction, although we send messages only in a single direction, the data passed in the last messages that were received before the change in direction is used for the calculation of the content of the following messages to be sent and for value assignment selections. Thus, after the first change of direction, all the constraints of the problem are considered.

6. MAX-SUM_AD WITH VALUE PROPAGATION

In this section we introduce value propagation into the Max-sum_AD algorithm. We start by presenting the motivation for this addition and then go into the algorithmic details.

6.1 Motivation for Value Propagation

In order to understand the need for value propagation in Max-sum in general and specifically in Max-sum_AD we identify two phenomena that deteriorate the ability of agents to identify the value assignments that will minimize the cost of the solution.

We illustrate the first phenomenon in the following example: The factor graph depicted in Figure 4 presents the standard need for value propagation as identified in previous studies on complete GDL algorithms [12, 18]. Each of the function-nodes computes for each of the values of its neighboring variable-nodes the minimal cost that it can offer by assigning a value to its other variable-node neighbor. In this specific symmetric example, each function identifies that for each value a, when assigning b to the other variable the cost is 0. Similarly for each value b, when assigning a to the other variable the cost is 0 as well. Thus, all messages sent by function-nodes to variable-nodes contain zeros for all values. As a result, messages from variable-nodes to function-nodes contain zero costs as well. In other words, in this specific problem, no information is propagated to the agents throughout the algorithm run. At the end of the run the variable-nodes are indifferent regarding their possible value assignments and if they all select value a the solution cost is 3. It is easy to see that there exists a solution in which one variable is assigned a and the other two assign b with a cost of 1.

While for complete algorithms ties are the only motivation for value propagation, the following example demonstrates that this is not the case in an incomplete Max-sum algorithm such as Max-sum_AD. Consider the factor graph depicted in Figure 5. Assume the order is according to the agents' indices. Function f_1 will send to x_3 costs 2 and 1 for its values a and b, respectively. Similarly, f_2 will send to x_2 costs 1 and 2 for its values a and b, respectively. Thus, the algorithm will converge to a solution that includes assignments $\langle x_2, a\rangle$ and $\langle x_3, b\rangle$. However, the selection of a for x_2 was under the assumption that value a is selected for x_1 while the selection of b for x_3 is under the assumption that x_1 is assigned b. Since only one of them can be assigned to x_1 the contribution of functions f_1 and f_2 to the solution cost is 6. If we would have selected first the assignment of x_1 and then the assignments for x_2 and x_3 accordingly, we could have reached a solution in which the contribution of those two functions was 3 (e.g., $\langle x_1, a\rangle$,$\langle x_2, a\rangle$ and $\langle x_3, a\rangle$). This inconsistency in the use of the information propagated by Max-sum_AD does not affect only the assignment selection. The information passed by messages is used to generate additional messages and therefore inconsistent information is propagated further to other agents in the distributed system. [3]

6.2 Introducing value propagation into Max-sum_AD

We overcome the pathologies we identified above by using a value propagation procedure similar to the method used in complete GDL algorithms for avoiding ties [12, 14, 18]. On iterations in which we perform value propagation we require that variable-nodes include in their messages to function-nodes their selected value assignments. Function-nodes select the best cost considering only the value assignments they received from their variable-node neighbors, which are ordered before them. Formally, in iterations in which we perform value propagation the message sent by function-node f to variable x includes as before for each $d \in D_x$:

$$min_{ass_{-x}} cost(\langle x, d\rangle, ass_{-x})$$

However, for a variable-node from which a value assignment was received in its latest message, the term $min_{ass_{-x}}$ considers only

[3]We demonstrate the phenomenon for Max-sum_AD since it is easier to follow. In standard Max-sum, such inconsistent information concerning the conflicting assignment of some node is propagated in all directions and through cycles, fed back to the node itself.

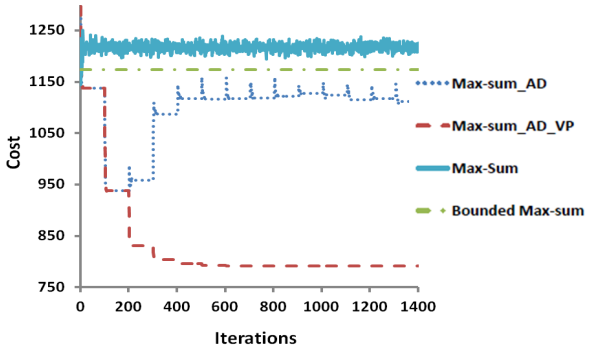


Figure 6: Solution cost of the Max-sum versions when solving problems with low density ($p_1 = 0.2$)

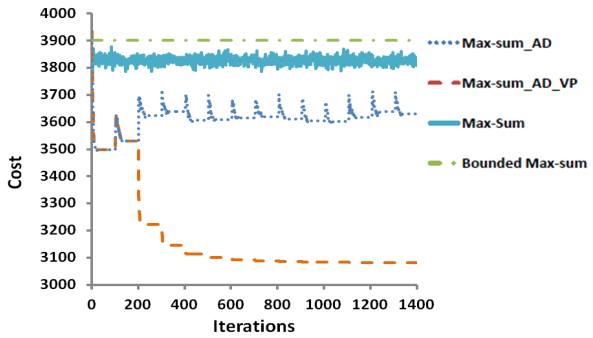


Figure 7: Solution cost when solving problems with high density ($p_1 = 0.6$)

this value assignment. Specifically, if a value assignment was received from each of the neighboring variable-nodes then $min_{ass_{-x}}$ is a single partial assignment.

In order to demonstrate the value propagation procedure, consider once again the factor graph depicted in Figure 5. Variable-node x_1 selects value a and includes this selection in the messages to function-nodes f_1 and f_2. Then f_1 calculates for the values of x_3 the costs 2 and 5 for values a and b, respectively. Similarly, function f_2 calculates costs 1 and 5 to the values a and b of variable-node x_2.

The timing for starting value propagation has a major effect on its success. If we would start value propagation from the first iteration, the sum of costs that indicate to agents which value assignments are better will not be propagated through the system. Thus, the selection of the first value assignment will be done in complete entropy and the following assignments can lead to an assignment with low quality. Instead, we start the value propagation procedure only after the algorithm converged in both directions (after the second order alternation). At this time agents have considered all the problem's constraints and have enough knowledge to make a quality selection of value assignments. Our experimental study also indicates that the best assignment found by Max-sum_AD without value propagation is after its second convergence (just before the second change of direction).

7. EXPERIMENTAL EVALUATION

In this section we present experiments that demonstrate the advantage of the proposed Max-sum_AD algorithm when combined with value propagation, over existing versions of the Max-sum algorithm.

The first set of experiments was performed on minimization random DCOPs in which each agent holds a single variable. Each variable had ten values in its domain. The network of constraints in each of the experiments was generated randomly by selecting the probability p_1 for a constraint among any pair of agents/variables. The cost of any pair of assignments of values to a constrained pair of variables was selected uniformly between 1 and 10. Such uniform random DCOPs with constraint networks of n variables, k values in each domain, a constraint density of p_1 and a bounded range of costs/utilities are commonly used in experimental evaluations of centralized and distributed algorithms for solving constraint optimization problems [8, 5].

The experimental setup included problems generated with 50 agents each. The factor graph generated for all versions of the Max-sum algorithm had agents performing the role of the variable-nodes representing their own variables, and for each constraint, we had the agent with the smaller index involved in it perform the role of the corresponding function-node.

We compared the Max-sum_AD algorithm with and without value propagation with the Max-sum and Bounded Max-sum algorithms [14]. We generated 50 random problems and ran the algorithms for 1400 iterations on each of them. The results we present are an average of those 50 runs. To make sure that the Max-sum_AD algorithms converge, we changed directions every 100 iterations, which is the longest possible path in the DAG (in case the graph has a chain structure). For each of the algorithms we present the sum of the costs of constraints included in the assignment it would have selected in each iteration.

Figure 6 presents the costs of the solutions found by the four algorithms when solving problems of relatively low density ($p_1 = 0.2$). It is most apparent that the Max-sum algorithm does not converge and it traverses complete assignments with high costs. The results of Bounded Max-sum are slightly better than the results of standard Max-sum. Max-sum_AD converges to a solution of lower cost even before the first direction change (in the first 100 iterations) and to a much better solution after the direction change. However, after the following direction changes it converges to solutions with higher costs. The solutions with the lowest costs are found by Max-sum_AD with value propagation. Since we begin the value propagation only after the second direction change, in the first 200 iterations it is identical to Max-sum_AD. However, after the second direction change it converges to a solution with a much smaller cost. It is interesting to mention that after the following direction changes it continues to improve until it finally converges to the solution with the lowest cost after the fifth direction change.

Figure 7 presents similar results for problems with higher constraint density – $p_1 = 0.6$. Here, it is notable that Bounded Max-sum finds solutions with higher costs than the standard Max-sum algorithm. This is reasonable since the more dense the problem is, the more edges are removed from the graph by Bounded Max-sum in order to produce the tree structured factor graph or, in other words, more constraints are ignored when producing the solution. The final difference in cost between the two versions of the Max-sum_AD algorithm seems similar to the results obtained for low density problems. However, here on dense problems, most of the reduction in cost by Max-sum_AD_VP was made after the second change of direction when the value propagation phase began. In addition, it is notable that the value propagation version of Max-sum_AD keeps improving even after 800 iterations (7 direction changes).

In our second set of experiments we present the relation between the costs of the solutions found by the different versions of the Max-sum algorithm and the optimal solution. Figures 8 and 9 present results of the four Max-sum versions solving smaller random problems on which we could run an exhaustive algorithm and

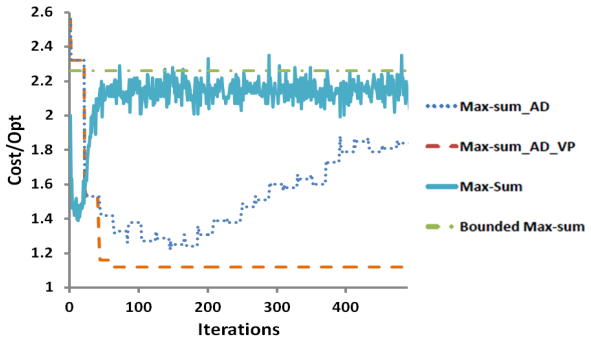


Figure 8: Solution cost when solving small problems with low density ($p_1 = 0.3$)

find the optimal solution. The problems included 10 agents, each holding a single variable with 5 values in its domain. Once again, we generated sparse and dense problems. However, the density parameters used were $p_1 = 0.3$ and $p_1 = 0.7$. This is because a lower density parameter for such small problems may generate problems with multiple components. On these small problems we could guarantee convergence by changing directions in Max-sum_AD every 20 iterations. Therefore, we ran the algorithms for a smaller number of iterations (500). The results presented include the factor from the optimal solution obtained by dividing the average cost of the assignments found by the algorithms in each iteration by the cost of the optimal solution.

It is interesting to observe that for both density parameters Max-sum_AD with value propagation found a solution with an average cost very close to the average optimal cost (1.12 factor for the sparse problems and 1.07 for the dense problems). It is also notable that Bounded Max-sum finds solutions with low quality for these problems (e.g., a factor larger than 2.5 of the average optimal cost for the dense problems). We assume this is caused by the smaller domains in these problems, which result in problems in which the constraints have larger differences among them. Thus, ignoring some of the constraints as done by Bounded Max-sum can have a larger effect on the result. It is interesting to notice the phenomenon that applies both to standard Max-sum and Max-sum_AD without value propagation. They both produce better results in the beginning of the run (i.e., in the early iterations of the algorithm) and then their performance deteriorates. This phenomenon is more apparent when solving low density problems. It is clear that the deterioration that these algorithms exhibit is prevented by value propagation. Thus, we assume that this deterioration is related to the propagation of inconsistent costs through the distributed system as we described in Section 6.1.

In the last set of experiments we generated graph coloring problems that include many ties. The problems we used included 50 agents, each holding a single variable and 3 colors in the domain of each variable. The density parameter was $p_1 = 0.05$, i.e., each agent had 2.5 neighbors on average. The problems are standard graph coloring problems, i.e., neighbors with identical colors induce a cost of 1 while for neighbors with different colors the cost is zero. The results presented in Figure 10 validate the observation we presented in Section 6.1 of Max-sum's weakness in the presence of ties. Both Max-sum and Max-sum_AD without value propagation are stuck with their initial assignment since no information is propagated through the system. On the other hand, Bounded Max-sum and Max-sum_AD_VP both include value propagation; therefore they are able to break the ties and produce high quality solutions. Max-sum_AD with VP still finds a solution with a smaller cost than Bounded Max-sum.

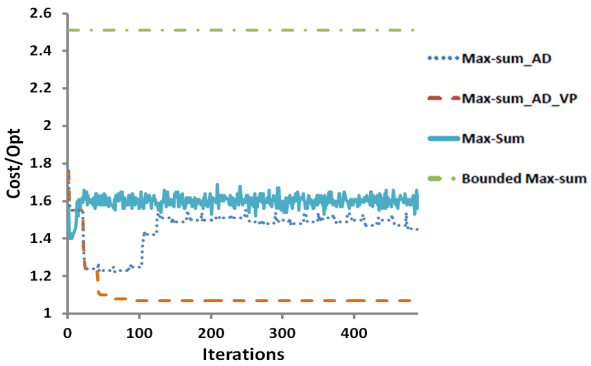


Figure 9: Solution cost when solving small problems with high density ($p_1 = 0.7$)

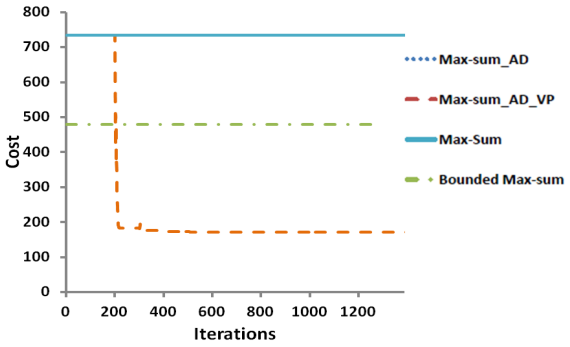


Figure 10: Solution cost when solving graph coloring problems

8. BOUNDED APPROXIMATION IN MAX-SUM_AD

While incomplete algorithms are not guaranteed to find the optimal solution, some of the recent studies on incomplete methods for DCOPs have offered quality guarantees and bounds on the distance of the produced solution from the optimal solution (e.g., [11, 14]). Specifically, *Bounded_Max-sum* [14] exploits the ability of Max-sum to find an optimal solution on a tree structured graph, and produces a bound from the optimal solution by reducing the problem to such a graph and accounting for the costs of the removed edges. This approach can be applied in Max-sum_AD as well. Using the messages of the Max-sum_AD algorithm we can dynamically generate a spanning tree of the factor graph and then run the algorithm on this tree in both directions to obtain the optimal solution.

In order to generate a spanning tree of the factor graph we need to identify cycles in the factor graph. To this end, we can carry the path (list of nodes) in a single direction on the algorithm's messages. When node n receives two messages in a single direction from two different neighbors that indicate that a node n' contributed to the costs calculation of both messages, it detects a cycle. By removing one of the edges through which it received these messages the agent can eliminate the cycle. If all cycles are eliminated, the result will be a spanning tree that was found in linear time. Running Max-sum_AD in both directions on the generated tree and then using value propagation in an additional iteration would give us the optimal assignment for this tree structured graph. Accounting for removed edges as in [14] would result in a bounded approximation.

We note that the method described above is naive in its selection of removed edges and therefore is expected to give worse bounds than the bounds found by [14]. We leave for future work the in-

vestigation of the information that can be accumulated in the alternating process of Max-sum_AD that can result in producing tighter bounds.

9. CONCLUSION

The Max-sum algorithm offers an innovative approach for solving DCOPs. Unfortunately, when problems include cycles of various sizes in the factor graph, the algorithm does not converge and the states it visits are of low quality.

In this paper we proposed a new version of the Max-sum algorithm, Max-sum_AD, which guarantees convergence. Max-sum_AD uses an alternating DAG to avoid cycles. We proved that when the algorithm is performed in a single direction, it converges after a linear number of iterations. After performing a linear number of iterations in each direction the algorithm converges to a high quality solution after considering all of the problem's constraints. If we keep alternating directions the algorithm converges to states that are not necessarily monotonically improving. In fact, our empirical results reveal that after the second direction change, the algorithm explores states of lower quality.

We further identify a possible reason for this behavior of the algorithm. We demonstrate that costs calculated by the algorithm propagated and used for selecting value assignments by agents are often considering different value assignment for the same variable and thus are inconsistent. In order to overcome this shortcoming of the algorithm we propose the use value propagation. To validate that we propagate values with high quality we begin the value propagation phase after the algorithm has converged for the second time and considered all the problem's constraints.

Our empirical study shows the advantage of the combination of Max-sum_AD with value propagation over previous versions of the Max-sum algorithm. Value propagation allows the algorithm to monotonically improve the solutions it finds in each direction until it converges to a solution with much higher quality than the other version of Max-sum find. On small problems for which we were able to find the optimal solution using a complete algorithm, Max-sum_AD_VP found solutions that approximate the optimal solution by a factor of roughly 1.1 on average.

Acknowledgment: We thank Arnon Netzer and Or Peri for producing the results of the complete algorithm and for their advice on how to improve the paper.

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, A. Meisels, P. Meseguer, and R. Zivan. Distributed constraint satisfaction with partially known constraints. *Constraints*, 14(2):199–234, 2009.

[3] I. Brito and P. Meseguer. Improving dpop with function filtering. In *AAMAS*, pages 141–148, 2010.

[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. 7th International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-08)*, pages 639–646, 2008.

[5] A. Gershman, A. Meisels, and R. Zivan. Asynchronous forward bounding. *J. of Artificial Intelligence Research*, 34:25–46, 2009.

[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] F. R. Kschischang and B. J. F. andH. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE TRANSACTIONS ON INFORMATION THEORY*, 47:2:181–208, Febuary 2001.

[8] J. Larrosa and T. Schiex. Solving weighted csp by maintaining arc consistency. *Artificial Intelligence*, 159:1–26, 2004.

[9] R. T. Maheswaran, J. P. Pearce, and M. Tambe. Distributed algorithms for dcop: A graphical-game-based approach. In *Proc. Parallel and Distributed Computing Systems PDCS)*, pages 432–439, September 2004.

[10] P. J. Modi, W. Shen, M. Tambe, and M. Yokoo. Adopt: asynchronous distributed constraints optimizationwith quality guarantees. *Artificial Intelligence*, 161:1-2:149–180, January 2005.

[11] J. P. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *IJCAI*, Hyderabad, India, January 2007.

[12] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *IJCAI*, pages 266–271, 2005.

[13] S. D. Ramchurn, A. Farinelli, K. S. Macarthur, and N. R. Jennings. Decentralized coordination in robocup rescue. *Comput. J.*, 53(9):1447–1461, 2010.

[14] A. Rogers, A. Farinelli, R. Stranders, and N. R. Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artif. Intell.*, 175(2):730–759, 2011.

[15] R. Stranders, A. Farinelli, A. Rogers, and N. R. Jennings. Decentralised coordination of continuously valued control parameters using the max-sum algorithm. In *AAMAS (1)*, pages 601–608, 2009.

[16] M. E. Taylor, M. Jain, Y. Jin, M. Yokoo, and M. Tambe. When should there be a "me" in "team"?: distributed multi-agent optimization under uncertainty. In *Proc. of the 9th conference on Autonomous Agents and Multi Agent Systems (AAMAS 2010)*, pages 109–116, May 2010.

[17] W. T. L. Teacy, A. Farinelli, N. J. Grabham, P. Padhy, A. Rogers, and N. R. Jennings. Max-sum decentralised coordination for sensor systems. In *AAMAS '08: Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, pages 1697–1698, 2008.

[18] M. Vinyals, J. A. Rodríguez-Aguilar, and J. Cerquides. Constructing a unifying theory of dynamic programming dcop algorithms via the generalized distributive law. *Autonomous Agents and Multi-Agent Systems*, 22(3):439–464, 2011.

[19] M. Vinyals, E. Shieh, J. Cerquides, J. A. Rodriguez-Aguilar, Z. Yin, M. Tambe, and E. Bowring. Quality guarantees for region optimal dcop algorithms. In *Proc. of 10th Int. Conf. on Autonomous Agentsand Multiagent Systems (AAMAS 2011)*, pages 133–140, Tapei, 2011.

[20] W. Yeoh, A. Felner, and S. Koenig. Bnb-adopt: An asynchronous branch-and-bound dcop algorithm. *J. Artif. Intell. Res. (JAIR)*, 38:85–133, 2010.

[21] W. Zhang, Z. Xing, G. Wang, and L. Wittenburg. Distributed stochastic search and distributed breakout: properties, comparishon and applications to constraints optimization problems in sensor networks. *Artificial Intelligence*, 161:1-2:55–88, January 2005.

Improving BnB-ADOPT$^+$-AC

Patricia Gutierrez Pedro Meseguer
IIIA - CSIC
Universitat Autonoma de Barcelona
08193 Bellaterra, Spain
{patricia|pedro}@iiia.csic.es

ABSTRACT

Several multiagent tasks can be formulated and solved as DCOPs. BnB-ADOPT$^+$-AC is one of the most efficient algorithms for optimal DCOP solving. It is based on BnB-ADOPT, removing redundant messages and maintaining soft arc consistency during search. In this paper, we present several improvements for this algorithm, namely (i) a better implementation, (ii) processing exactly simultaneous deletions, and (iii) searching on arc consistent cost functions. We present empirical results showing the benefits of these improvements on several benchmarks.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

distributed constraint optimization, soft arc consistency

1. INTRODUCTION

Distributed Constraint Optimization Problems (DCOPs) provide a useful framework for modeling many multiagent coordination tasks. Some of them are meeting scheduling [9], sensor network [6], traffic control [7], coalition structure generation [15], among others. DCOPs involve a number of distributed agents handling variables with finite domains and cost functions over positive integers. Agents exchange messages to coordinate and find a complete variable assignment with minimal cost.

Several distributed search algorithms have been proposed to optimally solve DCOPs: ADOPT [13], DPOP [14], NCBB [2], OptAPO [10], among others. In this paper we consider BnB-ADOPT [16], which uses depth-first branch-and-bound search. In particular, we work on the BnB-ADOPT$^+$-AC version [4], which combines BnB-ADOPT$^+$ with soft arc consistency (AC) in DCOP resolution. Soft arc consistency allows to calculate lower bounds that are useful to identify sub-optimal values, when the individual cost of a value surpasses a suitable upper bound. In BnB-ADOPT$^+$-AC sub-optimal values are removed dynamically during search, with two consequences. First, the search space of the problem becomes smaller, so its traversal can be done more efficiently. Secondly, as result of these deletions more informed lower bounds might appear, leading to further deletions. Although this process requires some extra computation and information exchange over the version not including AC, its overall effect is very beneficial for the global performance, leading to a substantial decrement in the amount of communication and computation required for optimal DCOP solving [4].

In this paper we present several improvements for BnB-ADOPT$^+$-AC, namely (i) a better implementation (ii) searching on arc consistent cost functions and (iii) processing exactly simultaneous deletions. We present an empirical investigation on two benchmarks, first comparing BnB-ADOPT$^+$ with the DP2 heuristic [1], with AC and with the combination AC-DP2. Interestingly, results show that combining AC with DP2 (something theoretically possible) produces better results than any of them in isolation. Second, the proposed modifications are empirically evaluated. Their combination always obtains the best results in both communication and computation for all problems tested. For some cases, savings reach up to one order of magnitude.

The paper is structured as follows. In Section 2 we summarize some concepts needed in the rest of the paper. In Section 3 we present a better implementation of BnB-ADOPT$^+$-AC which substancially reduces computation. In Section 4 we describe the issue of simultaneous deletions, providing solutions involving extra messages. In Section 5 we explain preprocess and search process using AC cost functions. We experimentally evaluate these points in Section 6. Finally, we conclude the paper in Section 7.

2. PRELIMINARIES

2.1 DCOP

A *Distributed Constraint Optimization Problem (DCOP)* is defined by $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{A}, \alpha \rangle$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of variables; $\mathcal{D} = \{D_1, \ldots, D_n\}$ is the set of finite domains of $\mathcal{X}$, such that x_1 takes values in D_1,...,x_n takes values in D_n; $\mathcal{C}$ is a set of cost functions used to evaluate the costs of value assignments; $\mathcal{A} = \{a_1, \ldots, a_p\}$ is a set of agents and $\alpha : \mathcal{X} \rightarrow \mathcal{A}$ maps each variable to one agent. We use binary $C_{ij} : D_i \times D_j \mapsto N \cup \{0, \infty\}$ and unary $C_i : D_i \mapsto N \cup \{0, \infty\}$ cost functions. The cost of a complete assignment, in which every variable has assigned a value, is the sum

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

of all binary and unary cost functions evaluated on those values. We make the common assumption that one agent handles only one variable, and thus we use the terms variable and agent interchangeably. Agents communicate through messages, which are never lost and are delivered in the order that they were sent. Message delay is random but finite.

A *DFS pseudo-tree* is a graph structure used to represent a DCOP instance, where nodes in the graph correspond to variables and edges connect pairs of variables appearing in the same binary cost function. There is a subset of edges called *tree edges* that form a rooted tree and are chosen following a depth-first (DFS) traversal of the graph. All other remaining edges are called *backedges*. Tree edges connect parent-child nodes, while backedges connect a node with its pseudo-parents and pseudo-children. Two variables sharing cost functions are in the same branch of the DFS tree. Several distributed algorithms exploit a pseudo-tree arrangement of their variables [13, 16], which allow agents positioned in different branches of the DFS to perform search in parallel.

2.2 BnB-ADOPT

BnB-ADOPT [16] is a distributed search algorithm that optimally solves DCOP. It is a depth-first branch-and-bound version of ADOPT [13] showing better performance.

Agents in BnB-ADOPT are first arranged in a DFS pseudo-tree, so they know their parent, pseudoparents, children and pseudo-children. Agents i and j sharing cost function C_{ij} maintain a local copy of the cost function. Every agent x_i maintain a *context* which contains its knowledge about the current value assignments of its ancestors. The context is updated through message exchange. For every domain value d and the current context, x_i maintains a lower and upper bound $LB(d)$ and $UB(d)$, and the bounds LB and UB, calculated in the following way:

$$\delta(d) = \sum_{(x_j,d_j)\in context} C_{ij}(d,d_j)$$

$$LB(d) = \delta(d) + \sum_{x_c\in children} lb_c(d) \qquad LB = \min_{d\in D_i}\{LB(d)\}$$

$$UB(d) = \delta(d) + \sum_{x_c\in children} ub_c(d) \qquad UB = \min_{d\in D_i}\{UB(d)\}$$

where $lb_c(d)$ and $ub_c(d)$ store the LB and UB values of children x_c for domain value d.

The goal of every agent is to explore and ultimately choose the value that minimizes LB. For pruning, agents store a threshold TH, which is an estimated upper bound calculated with the cost of the best solution found so far. The use of TH allows agents to change value more efficiently.

Some communication is needed to calculate the global cost of agents assignments and coordinate search towards the optimal solution. Three types of messages are used: VALUE, COST, and TERMINATE, with the following meaning:

- VALUE($i; j; val; th$): agent i informs child or pseudochild j that it has taken value val with threshold th;
- COST($k; j; context; lb; ub$): agent k informs parent j that with $context$ its bounds are lb and ub;
- TERMINATE($i; j$): agent i informs child j that i terminates.

Each agent executes the following loop: it reads and processes all incoming messages, decides about its value assignment and sends a VALUE message to each child or pseudochild and a COST message to its parent.

BnB-ADOPT$^+$ is a version of BnB-ADOPT which removes most of the redundant messages largely improving its efficiency, specially in communication. See [5] for details.

2.3 Soft Arc Consistency

Search can be improved by enforcing soft arc consistency, as a result some sub-optimal values can be identified and removed, making the search space smaller and therefore, speeding up the search process. Let (i, a) be agent x_i taking value a, $\top$ the lowest unacceptable cost and C_ϕ a zero-ary cost function that represents a lower bound of any complete assignment, we consider soft local consistencies defined in [8], as follows.

- *Node Consistency**: (i, a) is node consistent* (NC*) if $C_\phi + C_i(a) < \top$; x_i is NC* if all its values are NC* and there is $a \in D_i$ s.t. $C_i(a) = 0$; a problem is NC* if every variable is NC*.
- *Arc consistency**: (i, a) is arc consistency (AC) wrt. cost function C_{ij} if there is $b \in D_j$ s.t. $C_{ij}(a, b) = 0$; b is a *support* of a; x_i is AC if all its values are AC wrt. every binary cost function involving x_i; a problem is AC* if every variable is AC and NC*.

AC* can be reached modifying the original problem to obtain supports to NC* values and removing not NC* values. Supports are obtained on every value by (1) projecting the minimum cost from its binary cost functions to its unary costs, and (2) projecting the minimum unary cost into C_ϕ. Projection (1) requires the decrement of a minimum cost λ from the binary cost functions, and the increment of λ in the unary costs. Projection (2) requires the decrement of a minimum cost λ from the unary cost functions, and the increment of λ to C_ϕ. Every time a not NC* value is removed on agent x_i the AC* property must be rechecked on neighboring agents. The systematic application of these operations maintains the optimum in the resulting problem. These problems –the original and the AC* modified problem– are refered as *equivalent* [8].

NC* has become standard in the soft local consistency community, to the point that higher local consistencies using it are named without asterisk [3]. Following this trend, in the rest of the paper AC* is written AC.

2.4 BnB-ADOPT$^+$-AC

In [4] this algorithm is called BnB-ADOPT$^+$-AC*. Since, nowadays AC* is written AC [3], we follow this terminology and refer to this algorithm as BnB-ADOPT$^+$-AC (without asterisk).

BnB-ADOPT$^+$-AC [4] is an algorithm that combines distributed branch-and-bound search with soft arc consistency. Its search process is based on BnB-ADOPT$^+$, maintaining the same data and communication structure. The main difference is that agents are able to detect and delete sub-optimal values. A value can be found sub-optimal as result of enforcing AC on a copy of the original cost functions.

The inclusion of AC in BnB-ADOPT$^+$ have caused a number of modifications in the original algorithm, both in the structure of the exchanged messages and in the computation done. Regarding messages:

- COST messages include the variable *subtreeContr* that aggregates the costs of unary projections to C_ϕ made on every agent of the DFS subtree;
- VALUE messages include $\top$, which is constantly refined with the best solution found so far, and C_ϕ. Both are calculated at the *root* agent of the DFS;
- a new DEL message is introduced to inform deletions to neighbors; with message DEL$(i; j; d)$, i informs neighbor j that it has deleted value d. When received, neighbors recheck the AC property on their values, which may lead to further deletions.

Regarding memory, the domain of neighbors have to be represented in each agent, so memory requirements increase in $O(nd)$. Regarding computation, values are tested for deletion and cost functions are projected (binary into unary, unary into C_ϕ). A value d is proved sub-optimal and can be deleted unconditionally from the domain of x_i if $C_i(d) + C_\phi > \top$. Only the agent owner of a variable can delete values in its domain. AC enforcing is done in a preprocessing step and also during search every time a value is deleted.

When performing projections in two constrained agents i and j, changes on C_{ij} should be done carefully since i and j operate asynchronously and after a while they might end up with different copies of C_{ij}. In [4] authors explain how to maintain a *Legal Representation of Cost Functions*. To maintain a legal representation, i has to simulate the action of j on its C_{ij} copy, and vice versa. This can be done in the following way. There is a pre-established ordering for projections. In preprocess, agents always project first on higher agents and afterwards on lower agents (where higher/lower is referred to the position of agents in the DFS tree). When a deletion occurs in one agent, the agent projects binary costs over neighbors and sends DEL messages to neighbors. When a DEL message is received on a neighbor, the neighbor projects binary costs over $self$. By this, it is assured that the same cost is not counted twice when performing projections which may lead to delete optimal values. For details, see [4].

3. IMPROVING BNB-ADOPT$^+$-AC IMPLEMENTATION

In BnB-ADOPT$^+$-AC, agents check their domain every time there is a potential opportunity for deletion. For example, every time an agent processes a COST or a VALUE message, it checks if some values can be deleted from its domain. Since C_ϕ and $\top$ are informed in VALUE messages and the contribution of every children to C_ϕ is informed in COST messages, every time this information is updated a new opportunity for value deletion might appear. Now, we propose to check for deletions after the agent has completely processed the input queue. Such as it happens when agents decide to change value, agents will first gather all information from incoming messages before checking its domain. Also, instead of sending one DEL message for every deleted value, we send a list of all deleted values in the same DEL message. These modifications reduce computational cost and assure that agents will send at most one DEL message to each neighbour per cycle.

In BnB-ADOPT$^+$-AC unary projections to C_ϕ are done every time there is a possibility to increment the agents local contribution to C_ϕ. This can happen when a value is deleted or when AC is reinforced. Now, we propose to perform this operation after the agent has completely processed the input queue. We do this for the following reason. Every time there is a unary projection, the unary costs of the agent are decremented. In the centralized case, this decrement is quickly compensated with an increment in the global C_ϕ, but in a distributed setting this compensation is not immediate, it takes some time. First the agent contribution must travel in COST messages to the *root* agent, where is aggregated with other contributions. Afterwards the aggregated C_ϕ is informed in VALUE messages to lower neighbours. Since this process might take several cycles, we delay the decrement of unary costs on an agent until the next COST message is sent.

Delaying checking for deletions and projections of unary contributions to C_ϕ reduce considerably the computational effort made by BnB-ADOPT$^+$-AC, although in some cases this causes some extra messages (since these operations are not done as early as they could be). For empirical results on this see Section 6.

4. SIMULTANEOUS DELETIONS

If deletions are non-simultaneous in BnB-ADOPT$^+$-AC [4](that is, if two deletions never occur at the same time on neighboring agents), it is easy to see that projections are always done in the same order on every agent, so cost functions on both agents eventually remain equivalent. However, in the case that deletions occur at the same time on neighbouring agents, something different happens.

Consider the example in Figure 1. First row correspond to actions taking place inside agent i and second row actions taking place inside agent j. Every column show simultaneous operations, occurring at the same time on i and j. Agents i and j only store the unary costs of their own domain. Black domain values and costs are the actual values and costs stored in an agent. Gray domain values and costs are what an agent believes of the neighbor agent. Lines represent binary costs C_{ij} with cost one. Initially, $C_\phi = 0$, $C_i(b) = 1$, $C_j(b) = 1$ and the rest of unary costs are zero.

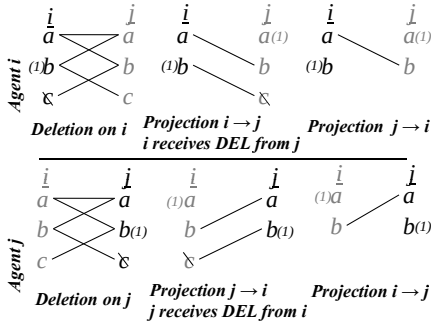


Figure 1: Agents i and j, and the process of two simultaneous deletions. Possible values for each agent are a, b, c, unary costs appear between parenthesis. In black, what an agent knows of itself. In grey, what an agent believes of the other agent. Lines indicate pairs of values with cost 1, no lines indicate cost 0.

On the first column, two simultaneous deletions take place. On second column, both agents make a projection over the neighbor. When projecting over the neighbor, binary costs are reduced from C_{ij} and the agent assumes that an increment in the unary costs of neighbor will eventually occur when the DEL message arrives to the neighbor. But notice that, because these operations occur at the same time, the order of resulting projections is opposite on agent i and j. This would not be the case if one deletion would have preceded the other. Then both agents would have kept the same ordering in projections (for example, a projection first from i to j and after from j to i) and they would have obtained $C_\phi = 1$. Notice that in the example C_ϕ remains zero.

Both agents projected at the same time a binary cost of 1 to the unary cost of its neighbor, but this operation has not actually taken place on the neighbor, so this cost has been lost from the problem. Notice that costs are not counted twice and no illegal deletions are produced, but we have lost a cost of 1 from the problem, which diminish deletion opportunities. In addition, on the last column the resulting cost functions on i and j are not equivalent. [1]

We can avoid this undesirable behavior assuring synchronous deletions. It is impossible that two agents know they are performing deletions at the same time, but it is possible that they communicate beforehand and agree on the order to follow. If one deletion always precedes the other, projections on neighboring agents maintain the same order. This assures that cost functions remain equivalent and no costs are lost from the problem.

4.1 Synchronizing Deletions

To maintain synchronous deletions, two main changes must be done in BnB-ADOPT$^+$-AC:

- Two new messages are introduced to synchronize deletions: SYNC_1 and SYNC_2
- Agents have a *locked* property. While an agent is *locked* it is able to read and process messages, but it will not change its value or send messages to neighbors, except for synchronization messages SYNC_1 and SYNC_2. An agent is *locked* because it is waiting to delete a value, or because a deletion is occurring in one or several neighbors. A *locked* agent changes to *unlocked* when it is no longer locked with any of its neighbors.

On Figure 2 a pseudocode of the synchronous deletion process for BnB-ADOPT$^+$-AC is shown. The rest of the algorithm is shown in Figure 3. The pseudocode is based on the implementation proposed in [4]. Modifications are described below.

Synchronizing deletion contains the following steps:

1. After completely processing the input queue, the `backtrack` method is invoked and the agent checks its domain looking for sub-optimal values [line 47].

[1]One may wonder if the approach of [4] is correct and complete. The answer is yes because search is done using a copy of the original cost functions which is only modified to reflect value deletions. There is another copy of cost functions used for AC enforcement, on which cost projections are done, but this copy is not used for search. This is further elaborated in section 5.

```
1  procedure CheckDomainForDeletions()
2    for each v ∈ D_self do
3      if C_self(v) + C_φ > ⊤ or (∑_{ch∈children} lb(ch, v) > ⊤
4      and childrenContexts(ch, v) = {self}) then
5        valuesToDelete.add(v);
6    if valuesToDelete.size > 0 then
7      for each k ∈ neighbors(self) do
8        if ¬hasStopped(k) then
9          sendMsg:(DEL, self, k, valuesToDelete);
10         locked(k) = true;
11     UpdateLockStatus();

12 procedure ProcessDelete(msg)
13   if locked(msg.sender) and self < sender then
14     processPending(msg.sender) = msg;
15   else
16     D_sender ← D_sender − {msg.valuesToDelete};
17     BinaryProjection(self, sender);
18     sendMsg:(SYNC_1, self, msg.sender);
19     if ¬hasStopped(msg.sender) then
20       locked(msg.sender) = true;
21       UpdateLockStatus();

22 procedure ProcessSYNC_1(msg)
23   locked(msg.sender) = false;
24   UpdateLockStatus();
25   if ¬locked then
26     D_self ← D_self − valuesToDelete;
27     valuesToDelete ← ∅
28     for each k ∈ neighbours do
29       BinaryProjection(k, self); sendMsg:(SYNC_2, self, k);
30     for each msg ∈ processPending do
31       ProcessDelete(msg);
32       processPending.remove(msg);

33 procedure ProcessSYNC_2(msg)
34   locked(msg.sender) = false;
35   UpdateLockStatus();

36 procedure UpdateLockStatus()
37   locked = false;
38   for each k ∈ neigbors(self) do if locked(k) then locked = true;

39 procedure ProcessStop(msg)
40   if msg.sender == parent then receivedTerminate ← true;
41   locked(msg.sender) = false;
42   UpdateLockStatus();
43   hasStopped(msg.seder) = true;

44 procedure Backtrack()
45   if locked then return;
46   UpdateLBUB();
47   CheckDomainForDeletions();
48   if locked then return;
49   if LB(value) ≥ min(TH, UB) then
50     value ← argmin_{v∈D_self}{LB(v)};
51   UnaryProjectionOverCo();
52   if value has changed then
53     SendValueToLowerNeighbors();
54   else
55     SendValueToChildrenToUpdateTH();
56   if (receivedTerminate or self == root) and LB == UB and
57   LB(value) == UB(value) then
58     SendStopMessageToLowerNeighbors();
59   SendCostToParent();
```

Figure 2: Pseudocode for Syncronizing Deletions.

2. A value v is proved sub-optimal under certain conditions [lines 3-4]: when its unary cost plus C_ϕ exceeds $\top$ or when the sum of its lower bounds exceeds $\top$ and this bounds were sent with a context that contains only the *self* agent (the bounds do not depend on any other agent, for more detail see [4]). When agent i realizes that it can delete values from its domain, instead of immediately erasing them, it marks them as pend-

```
60 procedure AC-Preprocess()
61   for each i ∈ neigbors(self) do
62     if i < self then
63       BinaryProjection(self, i);
64       BinaryProjection(i, self);
65     else
66       BinaryProjection(i, self);
67       BinaryProjection(self, i);
68   CheckDomainForDeletions();
69   UnaryProjectionOverCo();

70 procedure BinaryProjection(i, j)
71   for each a ∈ D_i do
72     v ← argmin_{b∈D_j}{C_ij(a, b)};
73     α ← C_ij(a, v);
74   for each b ∈ D_j do
75     C_ij(a, b) ← C_ij(a, b) − α;
76     if i = self then C_i(a) ← C_i(a) + α;

77 procedure Start()
78   for each d ∈ D_i, ch ∈ children(self) do
79       InitChild(ch, d);
80   InitSelf();
81   Backtrack();
82   loop forever
83     if (message queue is not empty) then
84       while (message queue is not empty) do
85         pop msg off message queue
84         Process(msg);
86       Backtrack();

87 procedure InitSelf()
88   value ← argmin_{v∈D_self}{LB(v)};
89   TH = ∞;

90 procedure InitChild(ch, d)
91   lb(ch, d) = 0;
92   ub(ch, d) = ∞;

93 procedure ProcessVALUE(msg)
94   add (msg.sender, msg.value) to context
95   CheckContextWithChildren();
96   if (msg.sender == parent) then
97     TH = msg.threshold;
98   if C_φ < msg.C_φ then C_φ = msg.C_φ;
99   if ⊤ < msg.⊤ then ⊤ = msg.⊤;

100 procedure ProcessCOST(msg)
101   d ← value of self in msg.context
102   PriorityMerge(context, msg.context);
103   CheckContextWithChildren();
104   if (context compatible with msg.context)
105     childrenContexts(msg.sender, d) = msg.context;
107     lb(msg.sender, d) = max{msg.lb, lb(msg.sender, d)};
108     ub(msg.sender, d) = min{msg.ub, ub(msg.sender, d)};
109   subtreeContr(msg.sender) = msg.subtreeContr;

110 procedure CheckContextWithChildren()
111   for each d ∈ D_i, ch ∈ children(self) do
112     if (childrenContexts(ch, d) incompatible with context) then
113       InitChild(ch, d);
```

Figure 3: Pseudocode for Preprocess and Process Phase.

ing to delete and sends DEL messages to neighbours $k_1, k_2, ..k_i$. Afterwards, i is *locked* with neighbours $k_1, k_2, ..k_i$, so i can process incoming messages but it can not change its value or send VALUE, COST or DEL messages [lines 6-11].

3. When neighbor k receives a DEL message from i it can be the case that k is already *locked* with i, this means that simultaneous deletions are taking place. In this case, the higher agent is the one that processes the DEL message first, otherwise the message remains as process pending [lines 13-14] and will be processed afterwards when the agent is *unlocked* [lines 30-32]. To process the DEL message, k deletes the values of i from its domain copy of i, and performs a projection $P_{i\to k}$ from i to k. After this, it sends a message SYNC_1 to i to inform that the deletion has been processed, and change its status to *locked* with i [lines 16-21].

4. Only after receiving SYNC_1 message from *all* its neighbours i is unlocked. At this point, all neighbours k have done a projection $P_{i\to k}$ from i to k. Then, i deletes the values from its domain, makes projections $P_{i\to k}$ on every neighbor k, and send a SYNC_2 messages to neighbours [lines 23-29].

5. When neighbour k receives a SYNC_2 message from i, it unlocks from i [lines 34-35].

6. A special case should be consider on termination. When an agent terminates execution it informs its lower neighbours [lines 40-43]. Once an agent has stopped, it will no longer be considered in the synchronizing process because it will not be able to respond, causing other agents to freeze forever. Therefore, before sending DEL messages to an agent or updating the *locked* status with an agent, it is first checked that the agent has not stopped execution [line 8, line 19].

5. SEARCH ON AC COST FUNCTIONS

One of the main goals of AC is to construct strong lower bounds. Zero-ary cost C_ϕ is a lower bound of the optimal solution. Unary cost $C_i(v) + C_\phi$ is a lower bound of domain value v. Lower bounds are useful to identify sub-optimal values when $C_i(v) + C_\phi > \top$. In addition, they can provide a heuristics for value selection which may improve search efficiency.

In [1] authors propose a preprocessing technique for ADOPT algorithm, and show that by calculating lower bounds for every domain value they are able to speed up ADOPT search. In [11] authors transform the original problem into an equivalent one projecting costs in a preprocessing step, then ADOPT is executed in the equivalent problem with performance improvements. In these two approaches authors aggregate lower bounds and use them during ADOPT search, but no deletions are performed.

We think deletions are a key point when enforcing soft arc consistency, since they lead to refinements in the lower bounds and further deletions. We perform deletions during AC preprocessing and also during search. Aiming at maintaining such deletions, we would like to use the same cost functions to perform search and to enforce AC: this would provide us a good value ordering (because costs are updated in the AC cost functions), combined with the deletions caused by AC enforcing.

Unfortunately, using the same cost functions for search and for AC enforcing is not an easy task. Consider the following case in Figure 4. Suppose an agent x has child ch, descendant d and ancestor a. There is a back-edge between a and d (Figure 4 (left)). Suppose a deletion takes place on descendant d and some costs c are moved from C_{da} to C_a (Figure 4 (center)). As result, an increment of costs occurs in a and a decrement occurs in ch subtree (this is because when calculating costs, agents aggregate their back-edges costs). The problem arises when COST messages arrive to

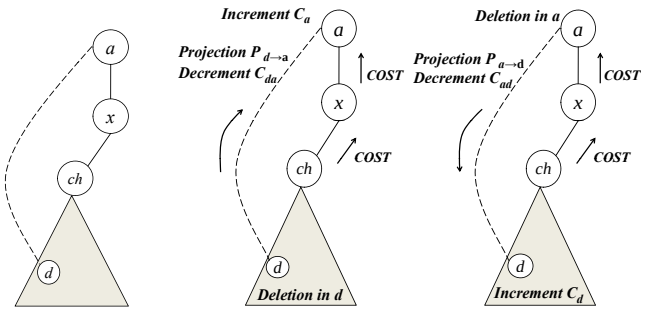


Figure 4: Performing projections during search process. Left: a problem instance. Center: a deletion takes place on a descendant. Right: a deletion takes place on an ancestor

x or a. If these messages where sent before deletion, they will contain in their LB, UB the cost c, but this cost is no longer in ch subtree, so the message must be ignored. Furthermore, the lb and ub tables in x and a should be reinitialized, because they may contain aggregated costs involving c, which will be counted twice if a COST message is sent from this agents without reinitialization.

Similarly, some problems occur if a deletion takes place on ancestor a and a cost c is moved from C_{ad} to C_d (Figure 4 (right)). There will be a cost increment in ch subtree and a decrement in a. If tables lb and ub are not reinitialized in x and a, the new COST messages from ch subtree containing c might not be accepted because the algorithm assumes that LB and UB improve monotonically, unless there is a context change.

Therefore, after deletions lb and ub tables must be reinitialized on neighbouring agent and even on other agents in the DFS branch. These reinitializations after deletions might lead to a serious degradation in performance, which makes deletion useless for efficiency gains.

To avoid reinitializations, we propose the following approach:

- Two copies of the cost functions are used: C_{search} and C_{AC}. Initially, they are identical.
- In preprocess, projections are performed on C_{AC}. Since we proposed a mechanism to synchronize deletions, it is assured that all projections are done in the same order on every agent, so no costs are lost from the problem and after preprocessing C_{search} and C_{AC} represent equivalent problems (one is AC, the other is not neccesarily AC).
- After preprocess, we make $C_{search} = C_{AC}$
- During search, costs are calculated using C_{search} aggregating binary, unary and zero-ary costs. Any new projection performed during search is done only on C_{AC}.
- During search, any value deletion computed using C_{AC} is applied on C_{search}. Observe that this is a legal operation.

Notice that syncronous deletions are needed during preprocessing to search in AC cost functions.

6. EXPERIMENTAL RESULTS

We evaluate experimentally the changes proposed in BnB-ADOPT$^+$-AC on two benchmarks: random DCOPs and structured Meeting Scheduling. Experiments are executed on a discrete event simulator and performance is evaluated in terms of the total number of messages exchanged among agents (#Total Msgs) and the number of non concurrent constraint checks (#NCCC) [12]. In every cycle of the simulator, agents read incoming messages from the message queue, process them and send outgoing messages. The simulation stops when all agents have stopped and no messages are exchanged.

Binary random DCOP are characterized by $< n; d; p_1 >$ where n is the number of variables, d is the domain size and p_1 is the network connectivity. A random instance contains $p_1 * n(n-1)/2$ cost functions. We tested random DCOP instances with: $< n = 10; d = 10; p_1 = 0.3, \ldots, 0.8 >$. Costs are selected from an uniform cost distribution. To introduce some variability among tuple costs, two types of binary cost functions are used, small and large. Small cost functions extract costs from the set $\{0, \ldots, 10\}$ while large ones extract costs from the set $\{0, \ldots, 1000\}$. The proportion of large cost functions is 1/4 of the total number of cost functions. Results appear in Table 1, averaged over 50 instances.

Meeting scheduling instances are obtained from the public DCOP repository [17]. In this formulation, variable represent meetings, domain represent time sots assigned for a meeting, and costs functions are set between meetings that share participants. We present cases A (23 variables), B (26 variables), C (71 variables) and D (72 variables). Results appear in Table 2, averaged over 30 instances.

We compare with several extensions of BnB-ADOPT. This algorithm has proved to be clearly more efficient than ADOPT and as efficient as NCBB for DCOP solving [16]. Comparison with other complete algorithm such as DPOP or OptAPO is truly difficult to measure –and scapes to the purpose of this paper. DPOP uses a linear number of messages but their size can be exponential, while BnB-ADOPT uses a linear size and exponential number of messages. OptAPO is a partially centralized algorithm while BnB-ADOPT is fully decentralized.

For every case in Meeting Scheduling and every network connectivity in random DCOPs we show results for:

1. First row: BnB-ADOPT$^+$ including a preprocessing step to calculate DP2 heuristic [1]. The preprocessing requires a single pass of messages from leafs to the *root* of the DFS tree calculating lower bounds for every value to focus search. This version improves over the basic BnB-ADOPT$^+$.
2. Second row: BnB-ADOPT$^+$-AC as described in [4].
3. Third row: BnB-ADOPT$^+$-AC with the DP2 preprocess.
4. Fourth row: BnB-ADOPT$^+$-AC with DP2 preprocess and the implementation improvements described in Section 3.
5. Fifth row: BnB-ADOPT$^+$-AC with DP2 preprocess, implementation improvements described in Section 3, synchronous deletions and searching on AC cost functions as described in Sections 4 and 5.

p_1	Algoritm BnB-ADOPT$^+$ with	#Total Msgs	#NCCC
	DP2	5,237	57,233
	AC	2,753	95,278
	AC-DP2	1,455	55,837
	AC-DP2-Opt	1,709	21,917
0.3	AC-DP2-Opt-Sync	**1,145**	**14,900**
	DP2	74,412	991,071
	AC	29,318	1,241,569
	AC-DP2	21,292	929,218
	AC-DP2-Opt	29,176	369,061
0.4	AC-DP2-Opt-Sync	**12,711**	**151,028**
	DP2	114,615	2,041,320
	AC	68,746	4,278,971
	AC-DP2	48,168	3,719,577
	AC-DP2-Opt	52,276	1,086,782
0.5	AC-DP2-Opt-Sync	**13,492**	**152,007**
	DP2	393,487	6,290,061
	AC	283,767	25,342,026
	AC-DP2	148,158	12,629,504
	AC-DP2-Opt	155,435	2,819,398
0.6	AC-DP2-Opt-Sync	**44,037**	**766,606**
	DP2	1,128,513	23,430,101
	AC	1,256,489	137,506,730
	AC-DP2	734,539	76,132,133
	AC-DP2-Opt	842,227	16,878,749
0.7	AC-DP2-Opt-Sync	**173,850**	**3,080,989**
	DP2	1,207,525	28,551,056
	AC	1,885,804	226,355,134
	AC-DP2	782,946	90,405,429
	AC-DP2-Opt	907,013	20,900,258
0.8	AC-DP2-Opt-Sync	**217,847**	**4,382,452**

Table 1: Experimental results on random DCOPs (10 variables) averaged over 50 instances.

When using DP2, communication and computational effort (#Total Msgs, #NCCCs) of DP2 preprocessing are included in the results. When searching on AC cost functions (fifth row), it is necessary to execute first the AC preprocess and afterwards the DP2 preprocess, so the bounds aggregated by DP2 correspond to the AC cost functions that will be used during search.

For random instances (Table 1), on small and medium connectivities, less messages are needed enforcing simple AC that using BnB-ADOPT$^+$ with DP2 (first and second row), on the other hand more NCCCs are needed since enforcing AC requires more computational effort. When combining BnB-ADOPT$^+$-AC with DP2 (third row), we observe a consistent decrement in messages and NCCCs. This confirms empirically that these techniques aiming to different objectives –the first to provide heuristic values during search, the second to erase sub-optimal values– can be enhanced when combined.

Optimizing the implementation of BnB-ADOPT$^+$-AC -(fourth row) combined with DP2 we obtain important reductions in NCCCs. We observe a moderate increment (10%-20%) in the number of messages. This effect is due to the slight delay in deletions and projections on C_ϕ (as mentioned in Section 3). However the decrement in computational effort is so important that globally we consider the modifications introduced as an improvement.

p_1	Algoritm BnB-ADOPT$^+$ with	#Total Msgs	#NCCC
	DP2	59,529	310,984
	AC	156,448	7,875,894
	AC-DP2	45,830	1,176,493
	AC-DP2-Opt	55,873	350,132
A	AC-DP2-Opt-Sync	**39,947**	**263,345**
	DP2	20,802	73,900
	AC	82,234	2,601,983
	AC-DP2	18,643	384,778
	AC-DP2-Opt	19,172	94,092
B	AC-DP2-Opt-Sync	**6,859**	**41,999**
	DP2	43,916	129,500
	AC	444,730	13,549,666
	AC-DP2	38,051	584,284
	AC-DP2-Opt	42,745	119,395
C	AC-DP2-Opt-Sync	**13,946**	**37,770**
	DP2	26,448	55,073
	AC	304,214	6,157,253
	AC-DP2	26,271	329,370
	AC-DP2-Opt	29,155	70,428
D	AC-DP2-Opt-Sync	**17,712**	**53,405**

Table 2: Experimental results on Meeting Scheduling instances (23, 26, 71 and 72 variables) averaged over 30 instances.

Including simultaneous deletions and searching in AC cost functions (fifth row) shows clear benefits in the number of exchanged messages and NCCC. Savings are higher on medium and higher connected problems, reaching up to one order of magnitude in some cases. This makes sense because on high connected problems, removing a sub-optimal value means avoiding context changes and reinitializations in many connected agents which are lower in the DFS tree.

In Meeting Scheduling problems (Table 2), we also observe important benefits. In these problem a simple preprocess to calculate DP2 heuristic is better than maintaining AC (first and second row). Moreover, we noticed during experimentation that the lower bound obtained in the *root* agent by DP2 preprocessing is actually very close to the optimal cost. This lead us to think that these instances, although contain a higher number of variables and cost functions than the random instances, are relatively easy to solve in the sense that only by DP2 preprocess we obtain a good estimation of the optimal cost, before starting the search. Observe that even in this case, our improved BnB-ADOPT$^+$-AC version (fifth row) is able to reduce messages and NCCC in all instances, in some cases by a factor of 2 or 3.

From these results we can extract some conclusions. First, it is beneficial to combined soft AC techniques with DP2 heuristic, the joint effort of both techniques is effectively summed-up and the result is an improvement in performance. Second, the combination of the proposed modifications causes substantial savings in both communication and computation effort with respect to existing versions of the considered algorithm. Third, maintaining soft AC to delete sub-optimal values provides more savings when the problem is connected and is hard to solve (the problem requires many messages and computation and the cost of the optimal solution can not be inferred accurately from a single pass preprocessing technique such as DP2).

7. CONCLUSIONS

In this paper we improve the algorithm BnB-ADOPT$^+$-AC, originally presented in [4], in several ways. First, we propose some modifications in the implementation of the algorithm, where checking for deletions and projections to C_ϕ are delayed until the agent executes the `Backtrack` procedure. Experimentally we show that this alternative reduces significantly the number of constraint checks, although the number of messages is slightly increased. This is due to the fact that an agent does not refine the C_ϕ or identify sub-optimal values as early as it could. However the decrement in computational effort is so important that we globally consider this change as an improvement.

Secondly, we address the issue of simultaneous deletions in the asynchronous setting of BnB-ADOPT$^+$-AC. When neighboring agents perform deletions at the same time, the order of projections in both agents is opposite and as a result some costs might be lost from the cost functions where AC is enforced. During search, BnB-ADOPT$^+$-AC uses original cost functions while AC is enforced in a copy of these cost functions, so the reported issue on simultaneous deletions does not affect optimality or termination. However by losing costs from the problem we lose information which could lead to identify sub-optimal values. To avoid this, we provide a synchronization mechanism to assure that projections are always done in the same order on every agent. This synchronization mechanism assures that no costs are lost but it requires some extra synchronization messages.

Finally, we propose to search on AC cost functions obtained after a preprocessing step since lower bounds calculated for every value can provide a heuristic for value selection. To do this, we need to assure synchronous deletions so $C_{original}$ and C_{AC} are equivalent after preprocessing (no costs are lost). Deletions are able to improve search and the inclusion of synchronization messages to guarantee that no costs are lost in the preprocessing, is compensated with message savings during search.

The aggregation of these three modifications produces a complete algorithm with communication and computation efforts substantially smaller than previous versions of BnB-ADOPT$^+$ (including either AC [4], DP2 heuristics [1] or a combination of both). In most cases, messages and NCCCs are reduced by at least a factor of 2, reaching up to one order of magnitude for some cases. These results allow us to consider the proposed approach as an important step foward towards more efficient algorithms for optimal DCOP solving.

It remains as future work to apply and evaluate the impact of the proposed techniques in other distributed search algorithms.

8. ACKNOWLEDGMENTS

We sincerely thank the anonymous reviewers for their comments, they helped us to make a better paper. Patricia Gutierrez has an FPI scholarship BES-2008-006653. She and Pedro Meseguer are partially supported by the project TIN2009-13591-C02-02.

9. REFERENCES

[1] S. Ali, S. Koenig, and M. Tambee. Preprocessing techniques for accelerating the DCOP algorithm ADOPT. *Proc. of AAMAS*, 2005.

[2] A. Chechetka and K. P. Sycara. No-commitment branch and bound search for distributed constraint optimization. In *Proc. of AAMAS*, pages 1427–1429, 2006.

[3] M. Cooper, S. de Givry, M. Sanchez, M. Zytnicki, and T. Werner. Soft arc consistency revisited. *Artificial Intelligence*, 174:449–478, 2010.

[4] P. Gutierrez and P. Meseguer. BnB-ADOPT$^+$ with several soft arc consistency levels. *Proc. of ECAI*, pages 67–72, 2010.

[5] P. Gutierrez and P. Meseguer. Saving messages in BnB-ADOPT. *Proc. of AAAI*, 2010.

[6] M. Jain, M. Taylor, M. Tambe, and M. Yokoo. DCOPs meet the realworld: Exploring unknown reward matrices with applications to mobile sensor networks. *Proc. of IJCAI*, 2009.

[7] R. Junges and A. L. C. Bazzan. Evaluating the performance of DCOP algorithms in a real world dynamic problem. *Proc. of AAMAS*, 2008.

[8] J. Larrosa and T. Schiex. In the quest of the best form of local consistency for weighted CSP. *Proc. of IJCAI*, 2003.

[9] R. Maheswaran, M. Tambe, E. Bowring, J. Pearce, and P. Varakantham. Taking DCOP to the real world: Efficient complete solutions for distributed event scheduling. *Proc. of AAMAS*, 2004.

[10] R. Mailler and V. Lesser. Asynchronous partial overlay: A new algorithm for solving distributed constraint satisfaction problems. *Journal of Artificial Intelligence Research*, 25:529–576, 2006.

[11] T. Matsui, M. Silaghi, K. Hirayama, M. Yokoo, and H. Matsuo. Directed soft arc consistency in pseudo trees. *Proc. of AAMAS*, 2009.

[12] A. Meisels, E. Kaplansky, Igor, Razgon, and R. Zivan. Comparing performance of distributed constraints processing algorithms. *Proc. of DCR*, pages 86–93, 2002.

[13] P. Modi, W.-M. Shen, M. Tambe, and M. Yokoo. ADOPT: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

[14] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proc. IJCAI-05*, pages 266–271, 2005.

[15] S. Ueda, A. Iwasaki, and M. Yokoo. Coalition structure generation based on distributed constraint optimization. *Proc. of 24th AAAI*, pages 197–203, 2010.

[16] W. Yeoh, A. Felner, and S. Koenig. BnB-ADOPT: Asynchronous branch-and-bound DCOP algorithm. *Journal of Artificial Intelligence Research*, 38:85–133, 2010.

[17] Z. Yin. USC DCOP repository. *http://teamcore.usc.edu/dcop*, 2008.

Optimal Decentralised Dispatch of Embedded Generation in the Smart Grid

Sam Miller, Sarvapali D. Ramchurn, Alex Rogers
Agents, Interaction and Complexity Group
School of Electronics and Computer Science
University of Southampton, UK
{sjom106,sdr,acr}@ecs.soton.ac.uk

ABSTRACT

Distribution network operators face a number of challenges; capacity constrained networks, and balancing electricity demand with generation from intermittent renewable resources. Thus, there is an increasing need for scalable approaches to facilitate optimal dispatch in the distribution network. To this end, we cast the optimal dispatch problem as a decentralised agent-based coordination problem and formalise it as a DCOP. We show how this can be decomposed as a factor graph and solved in a decentralised manner using algorithms based on the generalised distributive law; in particular, the max-sum algorithm. We go on to show that max-sum applied naïvely in this setting performs a large number of redundant computations. To address this issue, we present a novel decentralised message passing algorithm using dynamic programming that outperforms max-sum by pruning the search space. We empirically evaluate our algorithm using real distribution network data, showing that it outperforms (in terms of computational time and total size of messages sent) both a centralised approach, which uses IBM's ILOG CPLEX 12.2, and max-sum, for large networks.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*intelligent agents, multiagent systems*

General Terms

Algorithms, Theory, Performance

Keywords

DCOP, electricity, max-sum, coordination

1. INTRODUCTION

Due to recent incentives for cleaner electricity generation [13], there has been an increasing amount of renewable generators embedded in distribution networks [7, 9]. This poses a number of challenges for distribution network operators (DNOs).

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

Firstly, electricity networks are already highly capacity constrained; adding additional generation that is not managed effectively may overload the networks [14]. Secondly, it is much harder to balance electricity demand with generation from intermittent renewable resources. If the DNO fails to balance the supply and demand, the network can potentially become unstable which may result in brownouts, and in the worst case, cascading blackouts.

Thus, there is a clear incentive for DNOs to implement optimal dispatch[1] methods that are able to address these issues. That is, how should the generators be coordinated, such that the cost of the network is minimised (i.e., in terms of carbon dioxide (CO_2) emissions or generator running expenditure), whilst satisfying the loads and network constraints. Coordinating generators with respect to network cost, is known as active network management (ANM), and has recently been addressed in the power systems community [3, 14].

Within the ANM domain, a number of authors address the issues of coordinating generation from intermittent resources in the transmission network (where lines are less constrained than in the distribution network) [2, 8]. For example, Davidson et al. present an algorithm for changing the power outputs of the generators in the transmission network such that the cost of the network is minimised [2]. However, their technique involves a central authority calculating each generator's power output; who must have all the information about the entire network in order to calculate an optimal solution. As the size of the network grows, solving an optimisation problem in a centralised manner eventually becomes infeasible due to the exponential nature of the constraints [6].

In contrast, others have attempted to improve upon centralised approaches by decomposing the optimal dispatch problem and distributing the computation of its solution in order to improve its scalability [8, 10, 16]. For example, Kim and Baldick introduce a decentralised algorithm which uses Lagrangian techniques [8]. However, their algorithm has only been tested on problems containing up to two regions. Thus, it is unclear whether this approach could be applied to a large network. In the multi-agent systems literature, Kumar et al. introduce a message passing technique which extends distributed pseudotree optimisation procedure (DPOP) to solve the related area of research for reconfiguring feeder trees within a distribution network [10]. While this approach is decentralised, and was shown to work

[1]Optimal dispatch involves coordinating generators such that the loads and constraints of the network are satisfied.

on realistic sized networks, it does not address the problems highlighted above of incorporating an increasing amount of distributed generators (DGs) in the distribution network, and the need to coordinate their output.

Vytelingum et al. tackle the optimal dispatch problem by managing the trading of electricity between nodes on a network [16]. However, their technique has only been tested on problems containing up to 16 nodes. Thus, again it is unclear whether this approach could be applied to a larger network. Moreover, their technique is partially centralised; since each agent needs to know the entire network topology. In a large network, maintaining this system-wide knowledge is problematic, especially when faced with renewable generators whose output is continuously changing.

Against this background, in this paper we address the challenge of coordinating large numbers of renewable generators, embedded in the distribution network, by decomposing the optimal dispatch problem into a decentralised agent-based coordination problem; represented as a distributed constraint optimisation problem (DCOP). In more detail, each node in the network is represented by an agent which undertakes some of the computation required to solve the optimal dispatch problem; such that demands within the network are satisfied and CO_2 emissions of the entire network are minimised. We further decompose the DCOP as a factor graph and solve in a decentralised manner using algorithms based on the generalised distributive law (GDL) [1]; in particular, the max-sum algorithm [4]. We go on to show that max-sum applied naïvely in this setting performs a large number of redundant computations.

To address this issue, we present a novel message passing algorithm, called DYDOP (DYnamic programming Decentralised OPtimal dispatch), to calculate an optimal solution in a decentralised fashion. In particular, we solve the optimal dispatch problem on the most common distribution network types, namely radial networks, which tend to incorporate a high number of branches and sources [17]; for which centralised solutions scale poorly. Other common types of distribution networks include interconnected networks,[2] typically found in urban settings [5]. However, relays throughout the network ensure that all but one path is active at any one time; Multiple paths are present for security of supply. Therefore, our assumption of acyclic distribution networks throughout this paper is fully justified. Crucially, our algorithm handles the complexities of balancing flows within the network, *without needing central verification* of a particular solution. Thus, this paper makes the following contributions to the state of the art:

1. We provide a new formalism of the optimal dispatch problem as a DCOP. We show how this can be decomposed as a factor graph and solved using algorithms based on the GDL family, such as max-sum.

2. We present DYDOP, a novel decentralised message passing algorithm, that outperforms max-sum by only exploring the search space of feasible generator and distribution cable states.

3. We provide proof of the optimality of our algorithm and empirically evaluate it, on a large distribution network in India, showing that it outperforms (in terms of computational time and total size of messages sent) both a highly optimised centralised approach, which uses IBM's ILOG CPLEX 12.2, and max-sum.

When taken all together, our results set the benchmarks for the deployment of agent-based coordination algorithms to solve the optimal dispatch problem in the smart grid.

The remainder of this paper is organised as follows: Section 2 presents the formal model of the electricity network used for optimal dispatch. Section 3 details a new formalism of the optimal dispatch problem as a DCOP, and Section 4 shows how it can be solved by using max-sum. Section 5 presents our novel decentralised message passing algorithm DYDOP, presenting proof of optimality. Section 6 gives an empirical evaluation of DYDOP and Section 7 provides a discussion for future work. Finally, Section 8 concludes.

2. ELECTRICITY NETWORK MODEL

We now formally describe the model of an electricity network for which we need to solve the optimal dispatch problem. Hence, we consider the electric distribution network to be a network of generators, loads and distribution cables.

In more detail, we consider a set of n generators $\mathbf{G} = \{g_1, ..., g_n\}$. Each generator g_i has a certain discrete power output variable $\alpha_i \in S_i$ kW, where $S_i = \{s_1^i, ..., s_{q_i}^i\}$, $s_{q_i}^i \in \mathbb{R}^+$, $q_i \in \mathbb{Z}^+$ is the number of power output values for generator g_i, and $\mathbf{S}$ is an n-ary Cartesian product of S_i such that $\mathbf{S} = \{S_i \times ... \times S_n\}$. Let $\boldsymbol{\alpha} \in \mathbf{S}$ denote the set of power output variables for the generators in $\mathbf{G}$. Let $e_i = CI_i\alpha_i$ denote the CO_2 emissions that are produced when g_i, with carbon intensity $CI_i \in \mathbb{R}^+ \text{kgCO}_2/\text{kWh}$, outputs α_i.

We consider a set of m loads $\mathbf{L} = \{l_1, ..., l_m\}$. Each load l_i has a certain power consumption $\beta_i \in \mathbb{R}^-$ kW, where $\boldsymbol{\beta} = \{\beta_1, ..., \beta_m\}$ is the set of power consumption variables for the loads in $\mathbf{L}$.

We denote $\mathbf{V} = \{v_1, ..., v_k\}$ as the set of k nodes within the network. A node relays power to other nodes but can also contain a combination of generators and loads. Let $adj(v_i)$ denote all nodes that are connected to v_i via a distribution cable, let $\mathbf{L}(v_i)$ be the set of loads that are at v_i and $\mathbf{G}(v_i)$ be the set of generators that are at v_i.

$\mathbf{T}$ is the set of s distribution cables within the network, where $t_{ij} \in \mathbf{T}$ denotes the distribution cable between v_i and v_j. Each distribution cable has an associated thermal capacity $t_{ij}^c \in \mathbb{R}^+$ kW, which is the maximum power the cable can safely carry. It should be noted that we assume that all the distribution cables have the same reactance.

Finally, $\mathbf{W}(\mathbf{V}, \mathbf{T})$ is a finite undirected graph describing a network of nodes and distribution cables. $\mathbf{F}$ is the set of all power flows $f_{ij} \in \mathbb{R}$ kW, along the distribution cables in the network. Given the above definitions, the optimal dispatch problem, of finding an allocation of power outputs $\boldsymbol{\alpha}$, is defined as the problem of minimising:

$$\arg\min_{\boldsymbol{\alpha}} \sum_{i=0}^{n} CI_i\alpha_i \tag{1}$$

subject to the following constraints:

Constraint 1 The flow along a distribution cable cannot exceed its capacity:

$$|f_{ij}| \leq t_{ij}^c \tag{2}$$

[2]In an interconnected network there are multiple paths from a substation to a load.

Constraint 2 The net flow from v_i to v_j must be the opposite of the net flow from v_j to v_i:

$$f_{ij} = -f_{ji} \tag{3}$$

Constraint 3 The sum of the generators at v_i, the sum of the loads at v_i and the net flow from all nodes w connected to v_i is zero:

$$\sum_{w \in adj(v_i)} f_{wi} + \sum_{l \in \mathbf{L}(v_i)} \beta_l + \sum_{g \in \mathbf{G}(v_i)} \alpha_g = 0 \tag{4}$$

Having presented a model of the electricity network, the following section decomposes the problem into a DCOP.

3. DCOP REPRESENTATION

Formally, a DCOP is a tuple $\langle \mathbf{X}, \mathbf{D}, \mathbf{U} \rangle$ consisting of a set of h variables $\mathbf{X} = \{x_1, ..., x_h\}$ which can be assigned discrete values in the set of finite domains $\mathbf{D} = \{\mathbf{d}_1, ..., \mathbf{d}_h\}$ respectively. In our representation, $\mathbf{X} = \{\boldsymbol{\alpha}, \mathbf{F}\}$, with the domain:

$$\mathbf{d}_i = \begin{cases} S_i & \text{when } x_i \text{ is a generator} \\ \{-t^c_{ab}, ..., t^c_{ab}\} & \text{when } x_i \text{ corresponds to the distribution cable } t_{ab} \text{ between } v_a \text{ and } v_b \end{cases} \tag{5}$$

We note the set of relations as $\mathbf{U} = \{U_1, ..., U_k\}$ (also called utilities) where $U_i : \mathbf{d}_{i1} \times ... \times \mathbf{d}_{ij} \rightarrow \mathbb{R}^+$ denotes the cost of each possible combination of the involved variable values. We denote $\mathbf{A}$ to be the set of k agents. Each variable is assigned to an agent. Only the agent who is assigned the variable has knowledge of its domain and control over its value. Moreover, the utility U_i corresponds to the utility of agent i. In the context of the electricity network, U_i maps to the CO_2 emissions of v_i with respect to the constraints of the network (i.e., a lower cost means lower CO_2 emissions):

$$U_i = \begin{cases} \sum_{g \in \mathbf{G}(v_i)} CI_g \alpha_g & \text{if Equation (4) holds for } v_i \\ \infty & \text{otherwise} \end{cases} \tag{6}$$

where ∞ is used to penalise states that lead to inconsistent flows within the network (i.e., states that do not satisfy Equation 4).

With this in mind, it can be seen that the optimisation function for the electricity network, Equation (1), can be factorised in terms of the agent utility functions using Equation (6). The goal of the agents is to find an assignment $\mathbf{X}^*$ for the variables in $\mathbf{X}$ that minimises the sum of the costs:

$$\arg\min_{\mathbf{X}^*} \sum_{i=0}^{k} U_i \tag{7}$$

Typically, a DCOP can be represented by a factor graph, whose vertices correspond to variables and the edges denote the dependencies between the variables (i.e., the utility functions). Crucially, we provide a mapping of the DCOP to a factor graph that preserves the acyclic topology of the electricity network. Moreover, this mapping balances all of loads with generation, whilst satisfying the flow constraints of each distribution cable, and the constraints of the generators, in a fully decentralised way without needing centralised verification. Figure 1(a) shows an electricity distribution network consisting of distribution cables, generators and nodes. Example values for generator's maximum output, distribution cable's thermal capacity and power consumption at the loads are given. Node v_0 is connected to the rest of the electricity grid. Figure 1(b) shows the corresponding factor graph. By decomposing into a factor graph, the optimal dispatch problem can be solved using an algorithm from the GDL family, such as max-sum.

We choose max-sum to solve the DCOP because it maps directly onto a factor graph, and directly works with n-ary constraints (i.e., functions connected to more than two variables, see U_5 on Figure 1(b) for an example) without any additional modifications. Other algorithms which transform the DCOP into a depth first search (DFS) tree, such as ADOPT [11] and DPOP [12], suffer from scaling issues with the height and the width of the DFS tree respectively. Thus, max-sum is a natural fit to the optimal dispatch problem in a distribution network because networks of this nature often contain a large number of nodes and branches.

In max-sum, functions and variables can be arbitrarily assigned to any agent. However, in our model, each agent is assigned to compute one function which is associated to a specific node within the network. Moreover, a natural assignment of variables to agents involves an agent controlling the generator variables at its designated node, and the distribution cable variables connected to its node. If two or more agent's functions share the same variable, the variable is arbitrarily assigned to one of them. In Figure 1(b), the dashed circles give an example of the agents.

More importantly, since max-sum has been proven to converge to an optimal solution on acyclic factor graphs, and given that we provide a mapping from an acyclic electricity network to an acyclic factor graph, max-sum will be able to calculate the optimum solution to the optimal dispatch problem. The following section introduces the max-sum algorithm and explains how it can be applied to an electricity network.

4. MAX-SUM OPTIMAL DISPATCH

The max-sum algorithm (or min-sum as is the case with minimising CO_2 emissions) uses message passing in order to propagate the utilities of the variables around the factor graph. Messages are sent from variable to function, and from function to variable:

From variable to function:

$$Q_{b \rightarrow a}(x_b) = \sum_{a' \in A(b) \backslash a} R_{a' \rightarrow b}(x_b) \tag{8}$$

From function to variable:

$$R_{a \rightarrow b}(x_b) = \min_{\mathbf{X}_a \backslash b} \left[U_a(\mathbf{X}_a) + \sum_{b' \in B(a) \backslash b} Q_{b' \rightarrow a}(x_{b'}) \right] \tag{9}$$

where $B(a)$ is the set of variables connected to the function a, $A(b)$ is the set of functions connected to the variable x_b, and finally $\mathbf{X}_a \backslash b \equiv \{x_{b'} : b' \in B(a) \backslash b\}$.

A max-sum message being sent from function to distribution cable variable is a function of the flow in the cable with its domain bounded by the thermal capacity of the distribution cable. Consider the following example, shown in Figure 1(a). Let the distribution cable t_{59} between v_5 and v_9 have a thermal capacity t^c_{59} of 40kW, the load l_9 at v_3

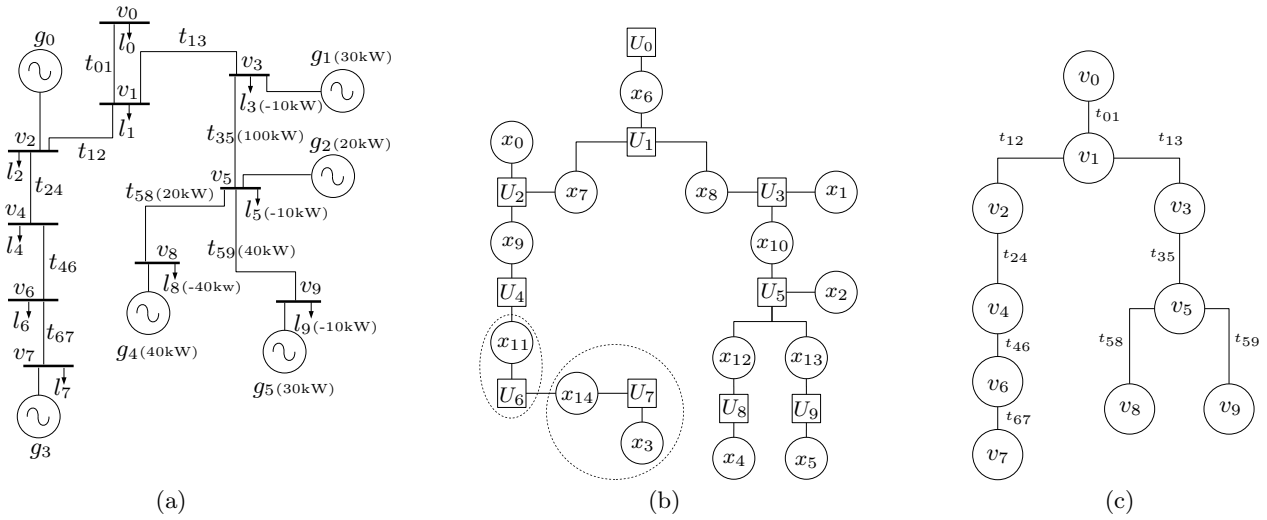


Figure 1: (a) An electricity distribution network. Showing example values for generator's maximum output, distribution cable's thermal capacity and power consumption at the loads. Node v_0 is connected to the rest of the electricity grid. (b) A factor graph representation of the same network. (c) The tree representation used by DYDOP.

be -10kW, and the generator g_5 at v_9 have a maximum output of 30kW. The message $R_{5\to13}(x_{13})$, sent from U_5 to x_{13} on the corresponding factor graph, Figure 1(b), will have domain $x_{13} \in \{-40, ..., 0, ..., 40\}$, having 81 utility values corresponding to the 81 variable states, discretised by 1kW steps. A +ve variable state indicates that the flow f_{59} is traveling from v_5 to v_9, and a -ve variable state indicates that f_{59} is traveling from v_9 to v_5.

A max-sum message being sent from function to generator variable is bounded by the maximum output of the generator. Consider the following example. Let the generator g_1 at v_3 have a maximum output of 30kW. The message $R_{3\to1}(x_1)$ will have domain $x_1 \in \{0, ..., 30\}$, having 31 utility values corresponding to the 31 variable states. Each state indicates the amount of power g_1 is producing α_1.

Messages are propagated around the factor graph until the values of the messages converge. Messages are guaranteed to converge to the optimal solution on acyclic graphs. At which point each variable chooses its optimal state based on the sum of the messages it has received:

$$Z_b(x_b) = \sum_{a \in A(b)} R_{a\to b}(x_b) \tag{10}$$

However, simply applying the max-sum algorithm naïvely in this manner produces poor performance. This is because much of the search space is infeasible and does not need to be searched. For instance, consider the previous example for the message $R_{5\to13}(x_{13})$. The message has a total of 81 variable states. However, the maximum amount of power that could travel along t_{59} from v_5 to v_9, in order to satisfy l_9, is only 10kW. Moreover, the maximum output of g_5 means that the maximum amount of power that could travel along t_{59} from v_9 to v_5, after l_9 is satisfied, is 20kW. Therefore, the utilities calculated for variable states $\{-40, ..., -21\}$ and $\{11, ..., 40\}$ are all infeasible. This highlights the wasted computation that a naïve implementation of max-sum performs. The domain of the message is bounded by t^c_{59}. However, the actual feasible states are dependant on the load and the available generation at v_9, which is considerably less. As the network size grows, this wasted computation becomes a major overhead (as we show in Section 5.1.2).

Thus, to address this issue, we present a novel decentralised message passing algorithm, DYDOP, which propagates messages from leaf nodes to the root of the tree network, such that only the utility of feasible states are calculated. As we show later, doing so greatly reduces the computation time as it allows us to prune much of the search space.

5. DYDOP OPTIMAL DISPATCH

We represent an acyclic electricity network as an acyclic network of nodes connected by distribution cables; Figure 1(c) shows the electricity network in Figure 1(a) transformed into this representation. DYDOP is applied to the acyclic network and uses a dynamic programming approach. Each node, which is controlled by an agent, has exactly one parent node and zero or more child nodes, apart from one node v_0 which is the root node and has no parent. Leaf nodes have no children, v_7, v_8 and v_9. Each node is assumed to have one or more generators, each with an associated carbon intensity, and one or more loads. DYDOP proceeds in two phases (which we describe in more detail in the following section):

Phase 1 – Value Calculation *PowerCost* messages are sent from the leaf nodes to the root node. A node waits until it has received *PowerCost* messages from all of its children before computing its own *PowerCost* message which it sends to its parent. Each *PowerCost* message describes the CO_2 emissions of its own generation and the generation of its children.

Phase 2 – Value Propagation When the root node receives *PowerCost* messages from all of its children, it calculates its optimum power output such that all

the demands of the network are satisfied and the CO_2 emissions are minimised. It then propagates power flow values to all its children which in turn propagate power flow values to their children.

The algorithm terminates when all leaf nodes receive a power flow value, at which point each node knows exactly what power it needs to output. We elaborate on the two phases below.

5.1 Phase 1: Value Calculation

In what follows we give a detailed overview of the DYDOP's value calculation phase. Section 5.1.1 introduces the structure of a *PowerCost* message, Section 5.1.2 describes how a leaf node constructs its *PowerCost* messages, and finally Section 5.1.3 details how a node merges its children's *PowerCost* messages.

5.1.1 *PowerCost* Messages

A *PowerCost* message sent from v_i to its parent $\hat{v_i}$, is an array of y *flowCO* elements:

$$PowerCost_{i\to\hat{i}} = [flowCO_1, ..., flowCO_y] \tag{11}$$

A *flowCO* element describes the CO_2 emissions that occur, when v_i and all of its children $chi(v_i)$ output certain amounts of power, such that there is a specified flow of power between v_i and its parent $\hat{v_i}$ along the distribution cable $t_{i\hat{i}}$:

$$flowCo_j =< f_{i\hat{i}}, \boldsymbol{\gamma}(f_{i\hat{i}}) > \tag{12}$$

where $f_{i\hat{i}} \in \mathbb{Z}$ kW is the resultant power flow travelling along $t_{i\hat{i}}$, and $|f_{i\hat{i}}| \leq t^c_{i\hat{i}}$ where $t^c_{i\hat{i}}$ is the thermal capacity of $t_{i\hat{i}}$. Note that $f_{i\hat{i}} > 0$ denotes the resulting power is flowing out of v_i to $\hat{v_i}$, $f_{i\hat{i}} < 0$ denotes the resulting power is flowing into v_i from $\hat{v_i}$, $f_{i\hat{i}} = 0$ denotes no power is flowing between v_i and $\hat{v_i}$. The function $\boldsymbol{\gamma} : \mathbb{R} \to \mathbb{R}^+$ kgCO_2/h denotes the CO_2 emissions that result from v_i and all of its children generating certain amounts of power.[3] Each *flowCO* element that v_i calculates maps to an *OPCState* which describes v_i's power output along with the *flowCO* elements of each of its children that results in the CO_2 emission described by the function $\boldsymbol{\gamma}$. This mapping represents the dynamic programming aspect of DYDOP since as power flow values are propagated down the tree, during the value propagation phase, the associated *OPCState* is used to find node v_i's power output given a particular power flow $f_{i\hat{i}}$.

5.1.2 Constructing a *PowerCost* Message at a leaf

Only the leaf node's power output needs to be taken into consideration when a leaf *PowerCost* message is constructed. For each power output v_i can produce, it constructs a corresponding *flowCO* element with flow $f_{i\hat{i}}$ calculated as:

$$f_{i\hat{i}} = \sum_{l\in\mathbf{L}(v_i)} \beta_l + \sum_{g\in\mathbf{G}(v_i)} \alpha_g \tag{13}$$

giving the resultant power flowing between v_i and $\hat{v_i}$. The CO_2 emissions $\boldsymbol{\gamma}$ of the *flowCO* element, is calculated as:

$$\boldsymbol{\gamma}(f_{i\hat{i}}) = \sum_{g\in\mathbf{G}(v_i)} \alpha_g CI_g \tag{14}$$

[3]Node v_9, Figure 1(c), with a carbon intensity of 0.3kgCO_2/kWh and a power output of 20kW, will have a resulting CO_2 emissions of 6kgCO_2/h and 10kW of resulting power travelling to v_5.

Algorithm 1 Constructing a leaf node *PowerCost* message

```
1. for αi ← 0 to genMax {
2.   rFlow ← αi + βi;
3.   rCO ← αi * CIi;
4.   flowCO(rFlow, rCO);
5.   linkToOPCState(flowCO, αi);
6. }
7. sendPowerCostMessageToParent();
```

where CI_g is the carbon intensity of generator g situated at v_i. See Algorithm 1 for a pseudocode representation of constructing a leaf node *PowerCost* message. We iterate through the generators different outputs, up to its maximum (line 1). For each output the resultant flow is calculated, (line 2) and the corresponding CO_2 emissions, (line 3). A *flowCO* element is created, (line 4), and then linked to the generators output which resulted in the resultant CO_2 emissions, (line 5). All the *flowCO* elements created are added to a *PowerCost* message and then sent to the nodes parent, (line 7). Note that the *OPCState*'s that are linked to by each *flowCO* element are never sent on to the parent node and are instead kept for use during phase 2 of the algorithm.

Consider the following $PowerCost_{9\to5}$ message, which v_9 sends to v_5, see Figure 1(a). Let the distribution cable t_{59} have a thermal capacity t^c_{59} of 40kW, the load l_9 be -10kW, the generator g_5 have a maximum output of 30kW and g_5 have a carbon intensity CI_5 of 0.1kgCO_2/kWh. The following is part of the $PowerCost_{9\to5}$ message:

$$\begin{array}{lcll} flowCo_j & = & <0, 1.0> & \to \quad [+10kW] \\ flowCo_{j+1} & = & <1, 1.1> & \to \quad [+11kW] \\ flowCo_{j+2} & = & <2, 1.2> & \to \quad [+12kW] \end{array}$$

Now, $flowCo_{j+2}$ indicates that a flow 2kW, from v_9 to v_5, will result in 1.2kgCO_2 emission with g_5 outputting 12kW. The total number of *flowCO* elements in $PowerCost_{9\to5}$ is 31. By contrast, compare with the example $R_{5\to13}(x_{13})$ message in Section 4, which has 81 variable states instead. This further highlights the wasted computation that the naïve implementation of max-sum performs.

5.1.3 Merging *PowerCost* messages

For each v_i that has at least one child, the *PowerCost* messages that it receives must be processed in order to produce its own *PowerCost* message that it sends to $\hat{v_i}$. The amount of power that can flow from v_i to $\hat{v_i}$, or from $\hat{v_i}$ to v_i, is bounded by $t^c_{i\hat{i}}$. With these bounds, v_i is able to calculate each valid flow that can travel into or out of it. For each valid flow, v_i calculates the minimum CO_2 emissions that result from v_i's output, and all of its children's outputs. To calculate the *flowCO* element for each resultant flow with the lowest CO_2 emissions value, v_i iterates through every possible power output that it can produce and every *flowCO* element from each of its children's *PowerCost* messages. A state represents the combination of one *flowCO* element from each of its children and v_i's power output. The flow $f_{i\hat{i}}$ of this state is calculated as:

$$f_{i\hat{i}} = \sum_{l\in\mathbf{L}(v_i)} \beta_l + \sum_{g\in\mathbf{G}(v_i)} \alpha_g + \sum_{c\in chi(v_i)} f_{ci} \tag{15}$$

Algorithm 2 Merging *PowerCost* messages

```
1. for αi ← 0 to genMax {
2.   foreach childPowerCost {
3.     rFlow ← αi + load + sum(OPCState);
4.     rCO ← (αi * CIi) + sum(OPCState);
5.     if(min(rFlow, rCO)) {
6.       PowerCost(rFlow, rCO);
7.       setNewMinimum(PowerCost);
8.       linkToOPCState(PowerCost, αi);
9.     }
10.  }
11.}
12.sendPowerCostMessageToParent();
```

where $\sum_{c \in chi(v_i)} f_{ci}$ is the sum of the chosen *flowCO* elements' flows from each of v_i's children. In order to choose the minimum state for each resultant flow, the CO_2 emissions of the state must be calculated as follows:

$$\boldsymbol{\gamma}(f_{i\hat{i}}) = \sum_{g \in \mathbf{G}(v_i)} \alpha_g CI_g + \sum_{c \in chi(v_i)} \gamma(f_{ci}) \qquad (16)$$

where $\sum_{c \in chi(v_i)} \gamma(f_{ci})$ is the sum of the chosen *flowCO* elements' CO_2 emissions from each of v_i's children. See Algorithm 2 for a pseudocode representation of merging *PowerCost* messages. We iterate through the generators different outputs, up to its maximum (line 1). For each output, we iterate through every possible combination of the *flowCO* elements from each of the its children's *PowerCost* messages (line 2). For a particular *OPCState* (i.e., a combination of *flowCO* elements, one from each child, and the generators output) the resultant flow is calculated by summing each flow of the *flowCO* elements, in the *OPCState*, with the generator output and the load, (line 3). Similarly, the resultant CO_2 emission is calculated by summing each CO_2 emission of the *flowCO* elements, in the *OPCState*, together with the product of the generators output and its carbon intensity, (line 4). If the resultant CO_2 emissions is the minimum recorded for the particular resultant flow, (line 5), then the *flowCO* element is created, (line 6), and set as the new minimum for that particular resultant flow, (line 7). The *flowCO* element is linked to the *OPCState*, (line 8). All the *flowCO* elements created are added to a *PowerCost* message and then sent to the nodes parent, (line 12).

As an example of merging *PowerCost* messages, consider the following $PowerCost_{5\to3}$ message, v_5 sends to v_3, see Figure 1(a). Let t^c_{35} be 100kW, t^c_{58} be 20kW, t^c_{59} be 40kW, l_5 be -10kW, l_8 be -40kW, l_9 be -10kW, g_2 have maximum output 20kW, CI_2 be 0.7kgCO_2/kWh, g_4 have maximum output 40kW, CI_4 be 0.25kgCO_2/kWh, g_5 have maximum output 30kW and CI_5 be 0.1kgCO_2/kWh. The following is part of the $PowerCost_{5\to3}$ message (after receiving messages from v_8 and v_9):

$$\begin{array}{lcccl} flowCO_j & = & < -10, 8 > & \to & [+0kW]8(-20)9(20) \\ flowCo_{j+1} & = & < -9, 8.25 > & \to & [+0kW]8(-19)9(20) \\ flowCo_{j+2} & = & < -8, 8.5 > & \to & [+0kW]8(-18)9(20) \end{array}$$

Now, $flowCo_{j+1}$ indicates that a flow of 9kW, from v_3 to v_5, will result in 8.25kgCO_2 emission with g_2 outputting 0kW, a flow 19kW from v_5 to v_8, and a flow 20kW from v_9 and v_5. The following section describes the second phase of DYDOP whereby power output values are propagated from the root node to the leaf nodes.

5.2 Phase 2: Value Propagation

Once the root node has received *PowerCost* messages from all of its children, it calculates how much power to output in order to satisfy all the loads within the network and minimise CO_2 emissions. It does this by iterating through every possible power output that it can produce and every *flowCO* element from each of its children's *PowerCost* messages. Equation (15) is used to calculate the resultant flow of a state. If the flow is not equal to zero, then this particular state for the network is infeasible, since excess flow means that supply and demand is imbalanced. For every state that has a flow equal to zero, the CO_2 emissions of the network are calculated by using equation (16).

The state with the minimum CO_2 emissions is selected as the optimum state of the network. Power flow values are then sent to each of the root node's children telling them which of their *flowCO* elements resulted in the minimum CO_2 emission. The child retrieves the correct *flowCO* element by matching the power flow value sent to them with the flow from the *flowCO* message. The *OPCState* which is referenced by each child recipient's corresponding *flowCO* element tells the child exactly how much power to output. The child recipient can then send the power flow of each *flowCO* element specified in the *OPCState* to each of its corresponding children. Power flow values are propagated in this manner to the leaf nodes, at which point each node in the network knows their optimum power output that results in the minimum CO_2 emissions for the entire network.

5.3 Completeness and Correctness

In what follows, we prove that DYDOP applied to trees is complete and correct:

PROPOSITION 1. *DYDOP is complete*

PROOF. *To construct PowerCost messages, v_i must iterate through all of its own possible generator outputs, S_i, and every flowCO element from each of its children's PowerCost messages. Each flowCO element contains the minimum CO_2 emissions that results from each $l \in \boldsymbol{L}(v_i)$, and all of its children's loads, being satisfied. The root node chooses a feasible state that results in the minimum CO_2 emissions. Therefore, at each node, all feasible states are evaluated and the root node chooses the optimal state which minimises CO_2. Hence, the algorithm is complete.* □

PROPOSITION 2. *DYDOP is correct*

PROOF. *This proof follows on from proposition 1. When constructing messages, v_i only evaluates feasible states; the states that conform to equations (2) – (4) and each $g \in \boldsymbol{G}(v_i)$'s maximum power output. Each message will contain the minimum CO_2 emissions that result from a feasible set of states. Therefore, any solution calculated by the algorithm will be valid as it has explicitly conformed to the constraints of the entire network. Hence, the algorithm is correct.* □

5.4 Computational Complexity

Here, the worst-case complexity of DYDOP is calculated, with regards to the size of the network and the number of

children a node has, in order to show its suitability for large optimal dispatch problems.

PROPOSITION 3. *The size of PowerCost messages that are sent by DYDOP grows linearly with the size of the network*

PROOF. *In the worst case, the maximum size of the message v_i has to create and send to $\hat{v}_i$ is Φ_i:*

$$\Phi_i = \frac{2t^c_{i\hat{i}}}{X_{\alpha_i}} \tag{17}$$

where $X_{\alpha_i} \in \mathbb{Z}^+$ is the discretisation of α_i and is currently 1; since each generator can produce power in 1 unit intervals. If generators are restricted to produce power in greater intervals, the size of the messages sent by each node can be reduced. In the worst case, the size of the messages DYDOP has to create and send in total is:

$$\sum_{v_i \in \boldsymbol{V} \setminus v_r} \Phi_i \tag{18}$$

where v_r is the root node. Therefore, the size of the messages DYDOP sends grows linearly in $O(|\boldsymbol{V}|)$. □

PROPOSITION 4. *The number of states that v_i must iterate through is exponential with $|chi_i|$*

PROOF. *When merging PowerCost messages, v_i must iterate through all states in the Cartesian product of all of its children's states and its own power output values. Therefore, the number of states a node must iterate through in the worst case grows exponentially in $O(M^{cmax})$, where cmax is the number of children a node has and M is the number of states a child has with a discretisation of X_{α_i}.* □

Even though the worst-case complexity of DYDOP is exponential in the number of children a node has, this is significantly less than the total number of nodes in the entire network. Thus, DYDOP may be able to exploit the structure of the network, unlike a centralised algorithm that does not explicitly take this structure into consideration, and compute an optimal solution faster. Therefore, the following section empirically evaluates DYDOP against a centralised approach and max-sum.

6. EMPIRICAL EVALUATION

In order to empirically evaluate DYDOP against max-sum and a centralised approach, we conducted an experiment on a real distribution network. The distribution network used is located in India and contains 76 substations;[4] the majority of the substations can further be connected to as many as 400 nodes. We only use one network because the topologies of distribution networks are largely similar. Our experiment was run in Java on a 2.67GHz Intel Xeon quadcore with 12GB of RAM, and was set up as follows. The number of additional nodes that could be connected to each substation was varied from 1 to 14, each with 50 iterations. During each iteration, nodes are assigned uniformly random loads, generators and carbon intensities. Each generator has 10 discrete power output levels and each distribution cable has its specified thermal capacity.

[4]A substation connects several distribution cables together and may contain generators, loads or transformers.

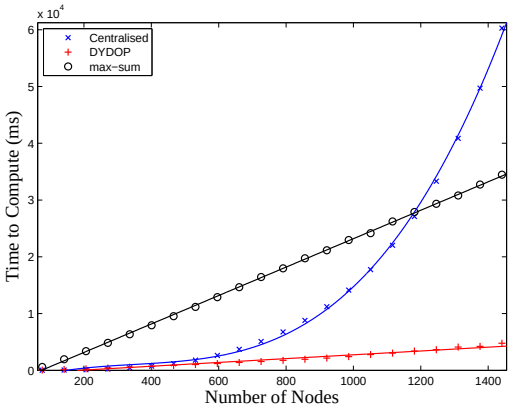


Figure 2: Time to compute a solution. India distribution network, 76 substations, varied number of nodes at each substation.

Figure 2 shows the computation time for the centralised approach, max-sum and DYDOP (error bars omitted due to being negligible). It can be seen that to start with, the centralised approach is as fast at computing a solution compared with DYDOP. However, after a network size of 460 nodes (which equates to only 6 additional nodes at each substation) DYDOP becomes significantly faster at computing a solution compared to the centralised approach. We used IBM's ILOG CPLEX 12.2 for the centralised approach which is highly optimised for solving optimisation problems.

Both DYDOP and max-sum's computation times increase linearly with the size of the network. This is because they both exploit the topology of the network. However, max-sum's computation time sharply increases compared with DYDOP. This highlights the unnecessary computation that max-sum is performing for infeasible variable states and shows the advantage of DYDOP. This is further highlighted in Figure 3 which shows that the total size of the messages sent using max-sum is much higher than DYDOP. Max-sum sends twice as many messages as DYDOP for the largest number of nodes we tested.

In contrast, the centralised computation time increases exponentially with the size of the network because it is unaware of the network structure, and seeks to solve the combinatorial optimisation by more standard approaches, such as the simplex method. Thus, as more DGs are added to distribution networks, it is clear that a centralised approach will quickly take an infeasible amount of time to compute a solution to the optimal dispatch problem.

In comparison, DYDOP is able to handle distribution networks with a large number of DGs and still calculate a solution in linear time. Therefore, our algorithm is a very good candidate for DNOs to use when solving future optimal dispatch problems in the ever growing distribution networks.

7. DISCUSSION

We believe DYDOP can be readily applied in many real-world electricity networks given the speed at which it resolves the generator outputs and the small amount of communication it requires. Particular applications include micro-grids with large numbers of small solar panels or micro-storage devices (on University campuses or military bases).

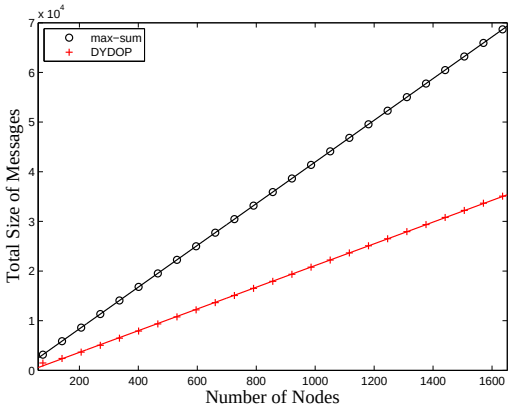


Figure 3: Sum total messages sent. India distribution network, 76 substations, varied number of nodes at each substation.

These applications typically involve network topologies that are either trees or radial and therefore match the type of network that DYDOP works on. Moreover, since the generators in these settings are typically low-power and discretise their power outputs (e.g. solar panels and batteries typically have set power outputs and can either be on or off), the assumptions we make about discretised generator outputs is perfectly valid in such settings.

Generalising our work to settings with non-discrete generator outputs will instead require handling continuous variables within DYDOP and it may be possible to extend some of the techniques introduced by [15] to do so. Moreover, to consider other distribution network topologies such as ring main,[5] we believe [18] can act as a starting point as they show that GDL algorithms can be made to converge on networks with a single loop.

8. CONCLUSIONS

In this paper we addressed the optimal dispatch challenges faced by DNOs. Namely how an increasing amount of cleaner DGs can be added to already highly constrained distribution networks, and coordinated in an efficient fashion using optimal dispatch. We provided a DCOP formulation of the optimal dispatch problem; we showed how this can be decomposed as a factor graph and solved in a decentralised manner using algorithms based on GDL; in particular, the max-sum algorithm. Furthermore, we showed that max-sum applied naïvely in this setting performs a large number of redundant computations.

To address this issue, we presented DYDOP, a novel decentralised message passing algorithm using dynamic programming, that outperforms max-sum by pruning the search space. It does this by propagating messages from leaf nodes to the root and only calculates the utility for feasible variable states. We empirically evaluated our algorithm using real distribution network data, showing that it outperformed (in terms of computational time and total size of messages sent) both a centralised approach and the max-sum approach for large networks.

[5]A ring main topology consists of a number of radial networks connected in a ring.

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. M. Davidson, M. J. Dolan, S. D. J. McArthur, and G. W. Ault. The use of constraint programming for the autonomous management of power flows. In *Proc. of the 15th Intl. Conf. on Intelligent System Applications to Power Systems*, pages 1–7, Curitiba, Brazil, 2009.

[3] Department for Business Enterprise and Regulatory Reform. Active network management (ANM) technology. Technical report, 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 *Proc. of the 7th Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 639–646, Estoril, Portugal, 2008.

[5] T. Gönen. *Electric power distribution system engineering*. McGraw-Hill New York, 2nd edition, 2007.

[6] M. E. Granada, M. J. Rider, J. R. S. Mantovani, and M. Shahidehpour. Multi-areas optimal reactive power flow. In *Proc. of the Transmission and Distribution Conf. and Exposition*, pages 1–6, Bogota, Colombia, 2008.

[7] N. Hatziargyriou, N. Jenkins, G. Strbac, J. A. Pecas Lopes, J. Ruela, and A. Engler. Microgrids-large scale integration of micro-generation to low voltage grids. *EU contract ENK5-CT-2002-00610, Technical annex*, 2002.

[8] B. H. Kim and R. Baldick. Coarse-grained distributed optimal power flow. *IEEE Transactions on Power Systems*, pages 932–939, 1997.

[9] J. K. Kok, M. J. J. Scheepers, and I. G. Kamphuis. Intelligence in electricity networks for embedding renewables and distributed generation. *Intelligent Infrastructures*, pages 179–209, 2010.

[10] A. Kumar, B. Faltings, and A. Petcu. Distributed constraint optimization with structured resource constraints. In *Proc. of the 8th Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 923–930, Budapest, Hungary, 2009.

[11] P. J. Modi, W. M. Shen, M. Tambe, and M. Yokoo. ADOPT: asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1-2):149–180, 2005.

[12] A. Petcu and B. Faltings. A scalable method for multiagent constraint optimization. In *Proc. of the 19th Intl. Joint Conf. on Artificial Intelligence*, pages 266–271, Edinburgh, Scotland, UK, 2005.

[13] S. D. Ramchurn, P. Vytelingum, A. Rogers, and N. R. Jennings. Putting the "smarts" into the smart grid: a grand challenge for artificial intelligence. *Communications of the ACM*, 2011.

[14] D. Roberts. Network management systems for active distribution networks: a feasibility study. *DTI Distributed Generation Programme (Contractor: SP PowerSystems LTD), Contract Number: K/EL/00310/00/00, URN*, 2004.

[15] T. Voice, R. Stranders, A. Rogers, and N. Jennings. A hybrid continuous max-sum algorithm for decentralised coordination. In *Proc. of the 19th European Conf. on Artificial Intelligence*, pages 61–66, Lisbon, Portugal, 2010.

[16] P. Vytelingum, S. D. Ramchurn, T. D. Voice, A. Rogers, and N. R. Jennings. Trading agents for the smart electricity grid. In *Proc. of the 9th Intl. Conf. on Autonomous Agents and Multiagent Systems*, pages 897–904, Toronto, Canada, 2010.

[17] B. M. Weedy and B. J. Cory. *Electric Power Systems*. John Wiley & Sons, 4th edition, 2004.

[18] Y. Weiss. Correctness of local probability propagation in graphical models with loops. *Neural computation*, 12(1):1–41, 2000.

DCOPs and Bandits: Exploration and Exploitation in Decentralised Coordination

Ruben Stranders, Long Tran-Thanh, Francesco M. Delle Fave, Alex Rogers &
Nicholas R. Jennings
University of Southampton
{rs2, ltt08r, fmdf08r,acr,nrj}@ecs.soton.ac.uk

ABSTRACT

Real life coordination problems are characterised by stochasticity and a lack of *a priori* knowledge about the interactions between agents. However, decentralised constraint optimisation problems (DCOPs), a widely adopted framework for modelling decentralised coordination tasks, assumes perfect knowledge of these factors, thus limiting its practical applicability. To address this shortcoming, we introduce the MAB–DCOP, in which the interactions between agents are modelled by multi-armed bandits (MABs). Unlike canonical DCOPs, a MAB–DCOP is not a single shot optimisation problem. Rather, it is a sequential one in which agents need to coordinate in order to strike a balance between acquiring knowledge about the *a priori* unknown and stochastic interactions (exploration), and taking the currently believed optimal joint action (exploitation), so as to maximise the cumulative global utility over a finite time horizon. We propose HEIST, the first asymptotically optimal algorithm for coordination under stochasticity and lack of prior knowledge. HEIST solves MAB–DCOPs in a decentralised fashion using a generalised distributive law (GDL) message passing phase to find the joint action with the highest upper confidence bound (UCB) on global utility. We demonstrate that HEIST outperforms other state of the art techniques from the MAB and DCOP literature by up to 1.5 orders of magnitude on MAB–DCOPs in experimental settings.

Categories and Subject Descriptors

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

General Terms

Algorithms, Theory, Experimentation

Keywords

Coordination, Distributed Problem Solving, Uncertainty

1. INTRODUCTION

Many real life applications can be modelled as systems of coordinating autonomous agents. Examples include wireless sensor networks (WSN), teams of UAVs deployed in disaster response scenarios and scheduling multi-processor jobs with unknown duration in distributed computing environments. These application domains often require decentralised solution techniques, as these yield scalable solutions, exhibit rapid response times and are robust to failure by avoiding the existence of a centralised point of control. As a result, building effective and efficient coordination algorithms has been a focus of attention within the multi-agent community.

This focus has led to the development of the framework of the distributed constraint optimisation problem (DCOP) [13]. In a DCOP, the agents' objective is to maximise a global utility function, that can be factorised into a sum of local utility functions that represent the interactions between agents. However, the canonical DCOP framework makes two strong assumptions that limit its practical applicability. First, the local utility functions are assumed to be known *a priori*. Second, it is assumed that the local utility functions are non-stochastic, i.e. that the same interaction always yields the same outcome. In reality, not only are utility functions typically noisy, they are also unknown before the agents' deployment. As an example, consider a WSN that is deployed by dropping sensor nodes from an aircraft. Since their position, neighbourhood and the environmental conditions are unknown before deployment, the utility functions that model the information gain received from their interactions need to be learnt online.

Crucially, in stochastic and *a priori* unknown environments, agents are no longer faced with a one shot optimisation problem as encoded in a canonical DCOP. Rather, we now have a problem in which agents have to coordinate to solve a sequence of "DCOP-like" problems in order to simultaneously reduce uncertainty about the local utility functions (*exploration*) and maximise the global utility (*exploitation*). This implies the need for striking a balance between exploration and exploitation. Focusing solely on exploration results in certainty about the agents' environment, but wastes resources by taking suboptimal actions. Similarly, consistently taking the joint action that is currently believed to be the best is also suboptimal because this belief might be incorrect.

To date, this challenge has not been satisfactorily addressed by the DCOP community; existing DCOP algorithms either do not consider the trade off between exploration and exploitation, or fail to make this trade off in a principled and efficient manner. Indeed, most existing DCOP algorithms, such as ADOPT, DPOP, and max-sum, are not able to represent either stochastic or unknown functions [13, 5, 12, 15, 14]. In particular, other local approximate algorithms have been proposed for problems with functions that are *a priori* unknown but non-stochastic [17, 6], or stochastic but with *a priori* knowledge about the un-

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

derlying distributions [2]. More recently, the E-DPOP algorithm has been proposed for solving DCOPs with utility functions whose values are influenced by exogenous sources of stochasticity. These sources are modelled by random variables, whose underlying probability distributions are either known [10] or unknown [9]. Whilst a significant contribution to the field, it divides exploration and exploitation into two distinct phases, a process that is known to make its performance dependent on the specific problem instance [16]. Moreover, as acknowledged by the authors, E-DPOP is an incomplete algorithm when applied to non-linear evaluation functions, and as a result can perform arbitrarily poorly on general problem instances [10].

In contrast, the multi-armed bandit (MAB) community has addressed the trade off between exploration and exploitation in a principled fashion from a single agent perspective [8, 3, 18]. A MAB is a simple analytical tool for modelling decision making under uncertainty. In more detail, a MAB is a slot machine with multiple arms, each of which yields a reward, drawn from an unknown but fixed probability distribution. The aim of the problem is to sequentially pull the arms so as to maximise the cumulative reward over a finite time horizon. To achieve this, a number of computationally efficient pulling algorithms have been proposed, such as ε-first [4], ε-greedy [16] and upper confidence bound (UCB) [3]. Whereas the former two are sensitive to the choice of the ε parameter, the latter provides optimal theoretical guarantees on the regret (the difference between its performance and that of the theoretical optimal solution) without any need for parameter tuning.

Thus, to address the shortcoming of canonical DCOPs, we develop HEIST[1], which combines the robustness and scalability of decentralised coordination with the optimal exploration/exploitation trade off that the MAB algorithms provide. HEIST solves MAB–DCOPs, an extension of canonical DCOPs, in which each local utility function becomes a MAB. This effectively models both the stochastic nature of realistic decentralised coordination problems, as well as the absence of *a priori* knowledge. Unlike a DCOP, a MAB–DCOP is not a single shot optimisation problem, but rather a sequential problem in which agents need to coordinate their joint actions over multiple time steps, so as to maximise the cumulative global utility received over a finite time horizon. HEIST achieves this by repeatedly choosing the joint action with the highest estimated upper confidence bound (UCB) on the sum of local utilities received in a single time step, which is a non-linear combination of the confidence bounds on the local utilities. HEIST optimally computes this joint action using a message passing algorithm known as generalised distributive law (GDL) [1]. The GDL algorithm has been shown to be very efficient for solving various factorisable optimisation problems, such as the one we face in this paper. By using the GDL to maximise the UCB in a decentralised fashion, HEIST is computationally efficient, and provides optimal asymptotic bounds on the regret of the global cumulative performance.

In more detail, this paper contributes to the state of the art as follows:

- We introduce MAB–DCOPs, a new formalism to represent decentralised coordination problems with stochasticity and the absence of *a priori* knowledge about local utility functions.
- We develop HEIST, a novel algorithm to solve MAB–DCOPs, and prove that it provides optimal asymptotic bounds on the regret of the global cumulative utility.
- We empirically evaluate HEIST in a reproducible controlled environment, and show that it outperforms other state of the art techniques from the MAB and DCOP literature (among which max-sum and ε-first) by up to 1.5 orders of magnitude on MAB–DCOPs.

The remainder of this paper is structured as follows. In Section 2 we discuss related work on MABs and DCOPs. In Section 3 we formally define MAB–DCOPs. In Section 4, we present HEIST, empirically evaluate it in Section 5, and conclude in Section 6.

2. PRELIMINARIES

As discussed in the introduction, our approach lies on the nexus of DCOPs and MABs. Therefore, in this section we discuss these two bodies of literature in more detail.

2.1 DCOPs

Decentralised coordination problems can be encoded as DCOPs, which are defined as follows:

DEFINITION 1. *A DCOP is a tuple $\langle A, X, D, U, F\rangle$ where:*

- $A = \{1, \ldots, |A|\}$ *is a set of agents.*
- $X = \{x_1, x_2, \ldots, x_n\}$ *is a set of variables.*
- $D = \{D_1, \ldots, D_n\}$ *is a set of finite domains, where D_i is the domain of variable x_i.*
- $F : X \rightarrow A$ *is a function that assigns variables to agents. Each agent controls (the value of) the variables that are assigned to it.*
- $U = \{U_1, \ldots, U_m\}$ *is a set of local utility functions defined over a local scope $\mathbf{x}_j \subseteq X$, which assigns a real value to each assignment to $\mathbf{x}_j$.*[2]

The solution of a DCOP is an assignment X^* to variables X that maximises the sum of the utility functions:

$$X^* = \arg\max_X \sum_{j=1}^{m} U_j(\mathbf{x}_j) \tag{1}$$

Many algorithms have been developed to solve DCOPs. Some of these [15, 14] exploit the generalised distributive law (GDL) to achieve computation and communication efficiency [1]. The GDL message passing algorithm exploits the factorisability of a broad class of optimisation problems (which includes DCOPs) in order to solve them in an efficient and decentralised manner. A defining property of these problems is that the valuation algebra of their global objective function is a *commutative semi-ring*, an algebraic structure which is defined as follows:

DEFINITION 2. *A commutative semi-ring is a triple $\langle R, \oplus, \otimes\rangle$, where R is a non-empty set and $\oplus$ and $\otimes$ are two (abstract) binary associative and commutative operators over R, such that $\otimes$ distributes over $\oplus$. Furthermore, there exists an identity element $\mathbf{0} \in R$ such that $x \oplus \mathbf{0} = x$ for all $x \in R$, and an identity element $\mathbf{1} \in R$ such that $x \otimes \mathbf{1} = x$ for all $x \in R$.*

[1] HEIST coordinates a group of bandits, hence its name.

[2] In the DCOP literature, these are also known as constraint functions.

The objective functions of the typology of problems solved by the GDL can be defined in terms of operators $\oplus$ and $\otimes$, a set of variables X and a set of functions U, similar to those in Definition 1. They can be encoded as *factor graphs* [7], undirected bipartite graphs in which vertices represent variables X and functions U, and edges encode the "is a parameter of" relation. The decentralised GDL message passing algorithm operates directly on a factor graph, and consists of two separate algorithms (Algorithms 1 and 2), one for each type of factor graph vertex. Algorithm 1 performs the computation associated with variables, and is executed by agent $F(x_i)$ that controls variable x_i, while Algorithm 2 performs the computation associated with functions, and is executed by one of the agents whose variables are a parameter of U_j.

Algorithm 1 The GDL algorithm for variable x_i. $\mathcal{M}(i)$ is the set of indices of neighbouring functions. $R_{j\to i}(x_i)$ is a message from function U_j computed in Algorithm 2

procedure GDL_VARIABLE(i)
  while stopping condition has not been met **do**
    for all $j \in \mathcal{M}(i)$ **do** ▷ For all adjacent functions
      if $|\mathcal{M}| = 1$ or no messages received yet **then**
        Send $Q_{i\to j}(x_i) = \mathbf{1}$ to U_j
      else
        Send $Q_{i\to j}(x_i) = \bigotimes_{k\in\mathcal{M}(i)\setminus j} R_{k\to i}(x_i)$ to U_j
      end if
    end for
    Wait for new messages from all $U_j : j \in \mathcal{M}(i)$
  end while
  return $Z_i(x_i) = \bigotimes_{j\in\mathcal{M}(i)} R_{j\to i}(x_i)$
end procedure

Algorithm 2 The GDL algorithm for function U_j. $\mathcal{N}(j)$ is the set of indices of neighbouring variables. $Q_{i\to j}(x_i)$ is a message from variable x_i computed in Algorithm 1.

procedure GDL_FUNCTION(j)
  while stopping condition has not been met **do**
    Wait for new messages from all $x_i : i \in \mathcal{N}(j)$
    for all $i \in \mathcal{N}(j)$ **do** ▷ For all adjacent variables
      Send to x_i message $R_{j\to i}(x_i) =$
$$\bigoplus_{\mathbf{x}_j\setminus x_i} \left(U_j(\mathbf{x}_j) \otimes \bigotimes_{k\in\mathcal{N}(j)\setminus i} Q_{k\to j}(x_k) \right)$$
    end for
  end while
end procedure

The following theorem is a fundamental property of the GDL message passing algorithm:

THEOREM 1. *If the factor graph is acyclic and the stopping criterion is chosen such that the algorithm is run for a number of iterations equal to the diameter of the factor graph, the following equation holds for each variable $x_i \in X$:*

$$Z_i(x_i) = \bigoplus_{X\setminus x_i} \bigotimes_{j=1}^{m} U_j(\mathbf{x}_j) \tag{2}$$

For proof of this theorem, see [11] (Chapter 26) and [1] (Theorem 3.1). The same result can be obtained for cyclic factor graphs by first transforming these into junction trees and using a slightly modified formulation of this algorithm [1]. For ease of exposition, in this paper we only consider acyclic factor graphs, in the knowledge that our algorithms and results also apply to junction trees with minimal modifications.

By instantiating the GDL algorithm for the max-sum commutative semi-ring $\langle \mathbb{R}, \max, + \rangle$, we obtain an algorithm for solving DCOPs (this algorithm is known as max-sum [15]). To see why this is the case, note that Equation 2 becomes:

$$Z_i(x_i) = \max_{X\setminus x_i} \sum_{j=1}^{m} U_j(\mathbf{x}_j) \tag{3}$$

This is the *maximum marginal* global utility that can be obtained for each assignment to variable x_i. As a direct consequence of this, setting $x_i^* = \arg\max_{x_i} Z_i(x_i)$ yields the variable assignment that maximises Equation 1. Note that this is only the case if the optimal solution is unique. If not, the solution can be made unique by adding small random values to the utility functions [15] (this is the method used in our experiments), or an additional utility propagation phase may be used [14]. We return to the GDL algorithm in Section 4 when we present HEIST, which uses the GDL algorithm instantiated for a special semi-ring designed to maximise the upper confidence bound on received utility.

2.2 Multi–Armed Bandits

A MAB is a slot machine with K arms, each of which delivers rewards drawn from an unknown distribution. The agent's goal is to choose which arms to pull so as to maximise expected cumulative reward over a finite time horizon T. In more detail, let P be a pulling policy (a sequence of pulls), and $i(t)$ denote arm chosen at time t by P, and $r_{i(t)}(t)$ the reward received by pulling that arm at time t. Then, we can formalise the agent's goal as follows:

$$P^* = \arg\max_{\mathbf{P}} \sum_{t=1}^{T} r_{i(t)}(t) \tag{4}$$

where $\mathbf{P}$ is the set of all possible policies. Clearly, if the reward distributions of each arm were known, the optimal policy would be to always pull the arm with the highest expected reward. This hypothetical scenario sets a performance benchmark known as *regret* against which to compare any policy. The regret $R_P(T)$ of a policy P after T time steps is the difference between the expected reward of the theoretical optimal policy and that obtained by P:

$$R_P(T) = \sum_{t=1}^{T} [\mu^* - \mu(i(t))] \tag{5}$$

where $\mu(i)$ is the *expected reward* of arm i and $\mu^* = \max_i \mu(i)$.

As mentioned in the introduction, there exist a number of pulling policies for minimising regret. Among these, UCB is one of the most widely used, since it is non-parametric and achieves asymptotically optimal regret. In more detail, UCB pulls each arm once at the beginning, then at each subsequent time step t, UCB selects arm $i^*(t)$ with the maximum upper confidence bound on the expected reward:

$$i^*(t) = \arg\max_{i\in[1,K]} \left[\hat{\mu}(i,t) + \sqrt{2 \ln t \frac{(u_{\max} - u_{\min})^2}{n(i,t)}} \right] \tag{6}$$

where $\hat{\mu}(i,t)$ is the sample mean of the rewards of arm i received until t, and $n(i,t)$ is the number of times UCB pulled arm i before time step t. UCB assumes the support of the reward function is bounded, i.e. $r_i(t) \in [u_{\min}, u_{\max}]$.

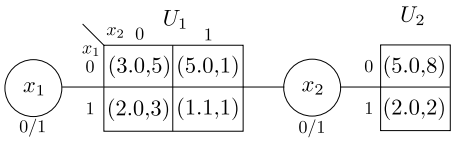


Figure 1: The MAB–DCOP from Example 1

The two terms in Equation 6 determine the trade off between exploration and exploitation. The larger the first, the more *exploitation* is favoured, since it is an estimate of the expected reward of arm i. The larger the second, the more *exploration* is favoured, since it represents the uncertainty in this estimate. In Section 4, we show how HEIST generalises the UCB algorithm to maximise the sum of rewards received from multiple (local) MABs in a MAB–DCOP and trades off exploration and exploitation in a decentralised fashion.

3. MAB–DCOPS

A MAB–DCOP is a DCOP where each utility function U_j is replaced by a MAB, such that each joint assignment $\mathbf{x}_j \in D_{\mathbf{x}_j}$ of the agents connected to that MAB becomes an arm of that bandit. Thus, in a MAB–DCOP, there is no *a priori* knowledge about the utility functions that govern the agents' interactions, and these interactions are subject to stochasticity. In more detail, for each $j \in \{1, \ldots, m\}$, the utility $U_j(\mathbf{x}_j)$ obtained by choosing $\mathbf{x}_j \in D_{\mathbf{x}_j}$ is drawn from an unknown, but fixed, distribution, with (unknown) expected value $\mu(\mathbf{x}_j)$, and bounded support $[u_{\min}, u_{\max}]$. The agents' goal is to choose an optimal joint assignment $X^*(t) = \langle x_1^*(t), \ldots, x_n^*(t)\rangle$ at each time t, such that the expected cumulative utility over a finite time horizon T is maximised:

$$[X^*(1), \ldots, X^*(T)] = \arg\max_{X(1),\ldots,X(T)} \mathbb{E}\left[\sum_{t=1}^{T}\sum_{j=1}^{m} U_j(\mathbf{x}_j(t))\right] = \arg\max_{X(1),\ldots,X(T)} \sum_{t=1}^{T}\sum_{j=1}^{m} \mu(\mathbf{x}_j(t)) \quad (7)$$

where $\mathbf{x}_j(t)$ is the chosen assignment to $\mathbf{x}_j$ at time t. To illustrate MAB–DCOPs, consider the following example.

EXAMPLE 1. *Consider the MAB–DCOP in Figure 1 with two binary variables x_1 and x_2, and two functions $U_1(\mathbf{x}_1)$ and $U_2(\mathbf{x}_2)$, at time $t = 10$. Utility functions are represented as tables, each cell of which has a tuple $(\hat{\mu}(\mathbf{x}_j, t), n(\mathbf{x}_j, t))$, where $\hat{\mu}(\mathbf{x}_j, t)$ is the sample mean of the utility received for the assignment $\mathbf{x}_j$ at t, and $n(\mathbf{x}_j, t)$ is the number of times that assignment has been sampled. Given the current sample means, assignment $(x_1 = 0, x_2 = 0)$ seems to be optimal. However, $(x_1 = 0, x_2 = 1)$ could be the real optimal, since $U_1(0, 1)$ and $U_2(1)$ have only been sampled at most twice.*

As this example suggests, to solve MAB–DCOPs, agents need to coordinate over the assignments to variables X at each time step, in order to trade off exploration (reducing uncertainty about the expected utility of each joint assignment) and exploitation (using the joint assignment that is believed to maximise reward).

Similar to MABs, we can define the regret of a coordination algorithm in a MAB–DCOP as a measure of the performance of a particular algorithm. In more detail, let $A = X^A(1), X^A(2), \ldots$ denote a coordination algorithm that chooses joint assignment $X^A(t) = \langle x_1^A(t), \ldots, x_n^A(t)\rangle$ at time t. The regret $R_A(T)$ for T time steps can be formalised as:

$$R_A(T) = T \cdot \max_X \sum_{j=1}^{m} \mu(\mathbf{x}_j) - \sum_{t=1}^{T}\sum_{j=1}^{m} \mu\left(\mathbf{x}_j^A(t)\right) \quad (8)$$

HEIST, which is described next, is an algorithm that enables agents to make the trade off between exploration and exploitation in a principled manner by provably achieving an asymptotically optimal regret.

4. THE HEIST ALGORITHM

In this section, we present HEIST, an algorithm for solving MAB–DCOPs with asymptotically optimal regret. Using HEIST, agents coordinate at each time step to identify the best joint assignment to variables X, i.e. the arms that should be pulled on the local MABs (functions U) to minimise regret over time horizon T. In more detail, at each time step t, HEIST uses a GDL message passing phase to find the joint assignment $X^*(t)$ that maximises the UCB on the received utility. The formulation of this UCB is different from the UCB given in Equation 6—in fact, it is a generalisation—since the objective in a MAB–DCOP is to maximise the sum of rewards of *multiple* local MABs, instead of a single MAB:

$$X^*(t) = \arg\max_X \left[\sum_{j=1}^{m} \hat{\mu}(\mathbf{x}_j, t) + \sqrt{2\ln t \sum_{j=1}^{m} \frac{(u_{\text{range}})^2}{n(\mathbf{x}_j, t)}}\right] \quad (9)$$

Here, $\hat{\mu}(\mathbf{x}_j, t)$ is the sample mean at time t of the utility obtained by assignment $\mathbf{x}_j$ from MAB U_j, $n(\mathbf{x}_j, t)$ is the number of times a specific assignment to $\mathbf{x}_j$ was made, and $u_{\text{range}} = u_{\max} - u_{\min}$.

EXAMPLE 2. *In Example 1, $X^*(10) = (x_1 = 0, x_2 = 1)$, with a UCB equal to $7 + \sqrt{2\ln(10)\left(1 + \frac{1}{2}\right)}$, assuming $u_{\text{range}} = 1$.*

Calculating joint action $X^*(t)$ in Equation 9 is not trivial. For instance, this problem cannot be solved using a DCOP algorithm, since the objective function is not decomposable into a sum of factors (i.e. it is non-linear). As a result, the application of a canonical DCOP algorithm to this problem can lead to sub-optimality [10] (which we will show in the empirical evaluation). In contrast, HEIST is optimal by applying GDL to the GDL–UCB semi-ring, a special semi-ring, which is guaranteed to preserve the joint assignment with the optimal UCB.

Now, at each time t, HEIST proceeds in two steps. First, it uses GDL instantiated for the GDL–UCB semi-ring to compute the *maximum marginal* UCB of each variable assignment — the maximum UCB that can be achieved for each assignment to an individual variable—in a decentralised fashion. Second, each agent uses the result of the first step to choose the variable assignments for the variables it controls that maximise the global UCB in Equation 9.

Before proceeding, with slight abuse of notation, we change the signature of local utility functions to output tuples of the form: $U_j(\mathbf{x}_j, t) = (\hat{\mu}(\mathbf{x}_j, t), b(\mathbf{x}_j, t)^2)$. Here, $\hat{\mu}(\mathbf{x}_j, t)$ is the sample mean of assignment $\mathbf{x}_j \in D_{\mathbf{x}_j}$ at time t and $b(\mathbf{x}_j, t) = u_{\text{range}}\sqrt{2\ln t / n(\mathbf{x}_j, t)}$ is its (local) upper confidence bound t (cf. the two terms in Equation 6).

HEIST is split up into two algorithms, one for variables (Algorithm 3) and one for functions (Algorithm 4). Before

Algorithm 3 The HEIST message passing algorithm for variable x_i

1: $t_{\min} := \max_{k\in[1,m]} |D_{\mathbf{x}_k}|$ ▷ Wait for initial samples (line 7)
2: **for** $t = t_{\min}+1$ to T **do** ▷ For each joint pull
3: $Z_i(x_i) = \text{GDL\_VARIABLE}(i)$ ▷ See Algorithm 1
4: Set $x_i^*(t) := \arg\max_{x_i} \left[\max_{(\hat{\mu},b^2)\in Z_i(x_i)} (\hat{\mu}+b) \right]$
5: Send message $\mathsf{sample}(x_i^*(t))$ to all $U_j : j \in \mathcal{M}(i)$
6: **end for**

Algorithm 4 The HEIST message passing algorithm for function U_j. Line numbering continued from Algorithm 3.

7: **for each** $\mathbf{x}_j \in D_{\mathbf{x}_j}$, sample $U_j(\mathbf{x}_j)$ once
8: $t_{\min} := \max_{k\in[1,m]} |D_{\mathbf{x}_k}|$
9: **for** $t = t_{\min}+1$ to T **do** ▷ For each joint pull
10: execute $\text{GDL\_FUNCTION}(j)$ ▷ See Algorithm 2
11: Wait for $\mathsf{sample}(x_i^*(t))$ messages from all $x_i : i \in \mathcal{N}(j)$
12: Pull arm $\mathbf{x}_j^*(t) = \{x_i^*(t) \mid i \in \mathcal{N}(j)\}$
13: Update sample mean $\hat{\mu}(\mathbf{x}_j^*, t)$ and sample count $n(\mathbf{x}_j^*, t)$
14: **end for**

communication commences, functions first sample the utility for their local domains (line 7) which takes $\max_{k\in[1,m]} |D_{\mathbf{x}_k}|$ time steps. This is analogous to the UCB algorithm, which pulls each arm once at the start. Then, functions and variables execute the GDL message passing algorithm (lines 2 and 8) instantiated for the GDL–UCB semi-ring, which is defined as follows:[3]

DEFINITION 3. *The GDL–UCB semi-ring is a semi-ring* $\langle R, \max_\succ, + \rangle$ *such that:*

- $R = \mathcal{P}(\mathbb{R}\times\mathbb{R})$ *a* set of sets[4] *of tuples, which have the same signature as the tuples output by functions* U_j*—the first element of each tuple is a sample mean* $\hat{\mu}$*, and the second is a squared bound* b^2*. The identity elements are* $\mathbf{0} = \{(-\infty,-\infty)\}$ *and* $\mathbf{1} = \{(0,0)\}$*.*
- $\max_\succ$ *is an operator that takes multiple sets* $S_1,\ldots,S_k \subseteq R$ *as input and outputs a set* S' *such that:*

$$S' = \max_\succ(S) = \left\{ s \in \bigcup_{i=1}^k S_i \mid \forall s' \in \bigcup_{i=1}^k S_i : s' \not\succ s \right\}$$

Operator $\max_\succ$ *filters out so-called* dominated *tuples from sets* $S_1,\ldots,S_k$*—those that cannot maximise the global UCB. Domination is formally defined by binary operator* $\succ$ *such that* $(\hat{\mu}_1, b_1^2) \succ (\hat{\mu}_2, b_2^2)$ *iff:*

$$(\hat{\mu}_1 - \hat{\mu}_2 > b_2 - b_1) \wedge$$
$$\left(\hat{\mu}_1 - \hat{\mu}_2 > \sqrt{b_2^2 + u_{\text{range}}^2 2\ln t} - \sqrt{b_1^2 + (u_{\text{range}})^2 2\ln t}\right) \wedge$$
$$\left(\hat{\mu}_1 - \hat{\mu}_2 > \sqrt{b_2^2 + u_{\text{range}}^2 \frac{2\ln t}{t}} - \sqrt{b_1^2 + (u_{\text{range}})^2 \frac{2\ln t}{t}}\right)$$

- $+$ *is a binary operator such that if* $S_1, S_2 \in R$*, then* $S_1 + S_2 = \{(\hat{\mu}_1+\hat{\mu}_2, b_1^2+b_2^2) \mid \forall(\hat{\mu}_1,b_1^2)\in S_1, \forall(\hat{\mu}_2,b_2^2)\in S_2\}$*. Thus,* $+$ *sums all pairs of tuples in* S_1 *and* S_2*.*

As a consequence of the non-linearity of the objective function in Equation 9, choosing the assignment that maximises the sum of UCBs on local utility, does not (necessarily) produce optimality of the UCB on the sum of local utilities. Thus, we cannot simply discard one tuple $(\hat{\mu}_1, b_1^2)$ in favour of tuple $(\hat{\mu}_2, b_2^2)$ if $\hat{\mu}_1 + b_1 < \hat{\mu}_2 + b_2$. This problem is addressed by the definition of the $\max_\succ$ operator, which is designed to discard only those tuples that are *guaranteed* to lead to global sub-optimality. As a result, it imposes a partial order over tuples to preserve the tuple that produces global optimality.

Executing the GDL algorithm on the GDL–UCB semi-ring yields the marginal function $Z_i(x_i)$ for each variable x_i (line 3). It can be proved that for each assignment $x_i \in D_i$, the set $Z_i(x_i)$ contains a tuple $(\hat{\mu}, b^2)$, such that $\hat{\mu}+b$ is the maximum achievable global UCB given that assignment (Theorem 2). Using this marginal function $Z_i(x_i)$, the second phase of HEIST first computes the maximum UCB for each assignment x_i (line 4, expression between brackets) and then selects the assignment with the maximum UCB (remainder of line 4). Then, in line 5, each variable informs adjacent functions of its assignment, after which all functions sample assignments (line 12).

EXAMPLE 3. *The following demonstrates the operation of* HEIST *on a single time step* ($t = 10$) *of the MAB–DCOP from Example 1. Let* $c = 2\ln(10)$*, and* $u_{\text{range}} = 1$*.*

GDL Iteration 1*:*

$$Q_{1\to1}(x_1) = Q_{2\to1}(x_2) = Q_{2\to2}(x_2) = \{(0,0)\}$$
$$R_{1\to1}(0) = \{(5,c)\}, \quad R_{1\to1}(1) = \{(2,c/3),(1.1,c)\}$$
$$R_{1\to2}(0) = \{(3,c/5)\}, \quad R_{1\to2}(1) = \{(5,c)\}$$
$$R_{2\to2}(0) = \{(5,c/8)\}, \quad R_{1\to2}(1) = \{(2,c/2)\}$$

GDL Iteration 2*:*

$$Q_{1\to1}(x_1) = (0,0), Q_{2\to1}(x_1) = R_{2\to2}(x_1), Q_{2\to2}(x_2) = R_{2\to1}(x_1)$$
$$R_{1\to1}(0) = \{(7,c(1+1/2))\}, \quad R_{1\to1}(1) = \{(7,c(1/3+1/8))\}$$
$$R_{1\to2}(0) = \{(3,c/5)\}, \quad R_{1\to2}(1) = \{(5,c)\}$$
$$R_{2\to2}(0) = \{(5,c/8)\}, \quad R_{1\to2}(1) = \{(2,c/2)\}$$

At this point, GDL has converged. We can now calculate the marginals $Z_i(x_i)$*:*

$$Z_1(0) = (7, 2\ln(10)(1+1/2)), \quad Z_1(1) = (7, 2\ln(10)(1/3+1/8))$$
$$Z_2(0) = (8, 2\ln(10)(1/5+1/8)), \quad Z_2(1) = (7, 2\ln(10)(1+1/2))$$

By calculating the UCB associated with these tuples, we obtain $x_1^* = 0, x_2^* = 1$*, with a UCB of* $7 + \sqrt{2\ln(10)\left(1+\frac{1}{2}\right)}$*, which indeed maximises Equation 9 (cf. Example 2).*

For the GDL message passing phase of HEIST, we can derive the following result:

THEOREM 2 (MAIN RESULT 1). *If the factor graph is acyclic and the stopping criterion is chosen such that GDL message passing phase is run for a number of iterations that is equal to the diameter of the factor graph, the following equation holds for each* $t > \max_{k\in[1,m]} |D_{\mathbf{x}_k}|$*:*

$$\max_{(\hat{\mu},b^2)\in Z_i(x_i)} (\hat{\mu}+b) = \max_{X\setminus x_i} \left[\sum_{j=1}^m \hat{\mu}(\mathbf{x}_j, t) + \sqrt{2\ln t \sum_{j=1}^m \frac{(u_{\text{range}})^2}{n(\mathbf{x}_j, t)}} \right]$$

Put differently, Theorem 2 states that, after the initial pulls, set $Z_i(x_i)$ contains the tuple that yields the *marginal maximum UCB* that can be obtained for each assignment to x_i, $i \in [1,n]$. As a direct consequence, line 12 pulls the arm on each function with the highest overall UCB (Equation 9). This observation leads to the following theorem:

[3]Proofs of correctness and explanation of the operators can be found in the Appendix.
[4]Here, $\mathcal{P}(S)$ denotes the powerset of set S.

THEOREM 3 (MAIN RESULT 2). *Suppose the factor graph is acyclic and the stopping criterion is chosen such that GDL message passing phase is run for a number of iterations that is equal to the diameter of the factor graph at every time t. Let $X^{\mathbb{E}} = \arg\max_X \sum_{j=1}^{m} \mu(\mathbf{x}_j)$ be the joint assignment that maximises the (unknown) expected utility. Let $d(X) = \mu\left(X^{\mathbb{E}}\right) - \mu(X)$ denote the difference between the expected utility of the optimal action $X^{\mathbb{E}}$ and that of a particular joint action X. Given this, the cumulative regret $R_{\text{HEIST}}(T)$ of* HEIST *after T time steps is at most:*

$$\sum_{X \neq X^{\mathbb{E}}} \frac{u_{\text{range}} 8 \ln T}{d(X)} + \left(1 + \frac{\pi^2}{3}\right) \sum_{X \neq X^{\mathbb{E}}} d(X) \qquad (10)$$

Based on this result, we can show that HEIST provides asymptotically optimal regret bounds, by comparing against best achievable regret:

THEOREM 4 (MAIN RESULT 3). *For any algorithm, there exists a constant $C \geq 0$, and a particular instance of the MAB–DCOP problem, such that the regret of that algorithm within that particular problem is at least $C \ln T$.*

Thus, the regret bound of HEIST (Equation 10) only differs from the best possible with a constant factor. The proofs of the theorems can be found in the Appendix.

5. EMPIRICAL EVALUATION

In the previous section, we proved that the regret achieved by HEIST is guaranteed to be a constant factor away from the optimal. However, further empirical analysis is needed to gauge HEIST's practical performance, in terms of solution quality as well as communication and computation overhead. Moreover, such analysis can focus on the algorithm's performance when the GDL phase of the algorithm is not run until convergence, one of the conditions for optimality in Theorem 3. Instead, the number of iterations c of the GDL phase can be a parameter for tuning the trade off between solution quality and overhead (i.e. computation and communication). Note that it is not the objective of these experiments to study the properties of MAB–DCOPs and HEIST across all possible probability distributions, and instantiations of the utility function U. Due to space constraints, we would not be able to do justice to the requirements of different application domains, and the specific configurations of HEIST. This is left for future work.

Therefore, in this section, we benchmark several versions of HEIST against existing approaches, taken from the state of the art in the MAB and DCOP literature. Specifically, we compare HEIST against the following algorithms:

HEIST-c a version of HEIST where the GDL message passing phase is run for c iterations. When taking joint actions is cheap compared to communication, or when action is required before the GDL message passing phase is able to converge, c can be set to a value smaller than the diameter of the factor graph. In this case, optimality is no longer guaranteed, but it can lead to a good trade off between communication and solution quality.

ε-first an algorithm that samples from the utility functions (exploration) for the first εT time steps, and picks the one that is believed to be optimal (exploitation) for the remaining $(1-\varepsilon)T$ time steps. At the start of the exploitation phase, this algorithm runs a standard DCOP algorithm (max-sum) once to find the joint assignment that maximises the sum of the sample means of the local utilities. Using a DCOP algorithm in this way is equivalent to using E-DPOP [9] on a problem where each local assignment $\mathbf{x}_j$ for $j \in [1, m]$ is modelled by a single random variable. Thus, ε-first can be considered as E-DPOP applied to a MAB–DCOP. To perform well, ε needs to be tuned for each problem instance [3, 16]. After initial tests, we found that $\varepsilon = 0.02$ leads to good performance for the problems described below.

Monolithic UCB the (centralised) UCB algorithm that considers a MAB–DCOP as a single "monolithic" MAB with $\prod_{i=1}^{n} |D_i|$ arms.

Max-Sum a standard DCOP algorithm, applied to a DCOP in which the objective is to compute $X^+(t)$ at every $t \in [1, T]$, such that:

$$X^+(t) = \arg\max_X \sum_{j=1}^{m} \left[\hat{\mu}(\mathbf{x}_j, t) + \sqrt{\frac{(u_{\text{range}})^2 2 \ln t}{n(\mathbf{x}_j, t)}}\right] \qquad (11)$$

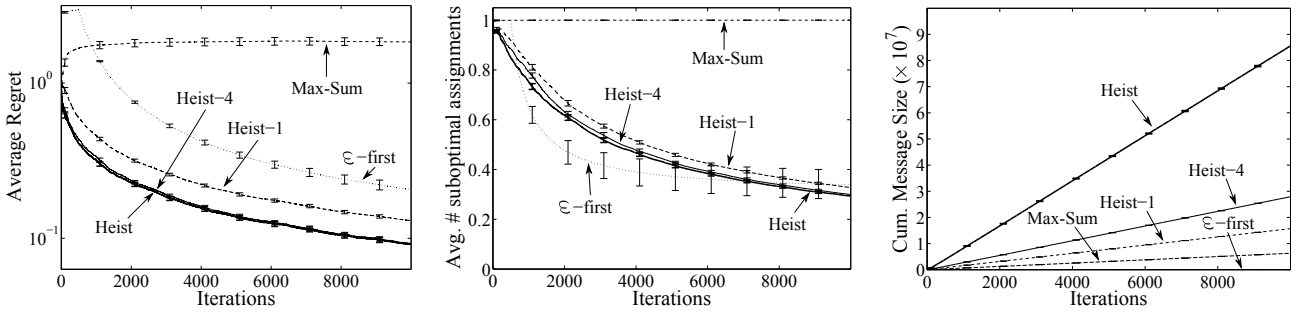
By solving this DCOP, the sum of UCBs on local utilities is maximised, instead of the UCB on the sum of utilities (Equation 9). Since the latter is clearly not linear, Equations 9 and 11 are not equivalent (except for $m = 1$). As a result, this decomposition leads to loss of optimality in the sense of Theorem 4. Léauté et al. observed a similar result for their (incomplete) algorithm when using a linear decomposition to maximise non-linear objective functions [10].

The ε-first and max-sum algorithms are included to demonstrate that standard DCOP algorithms are unsuitable for solving MAB–DCOPs, while the Monolithic UCB algorithm is included to demonstrate the need for exploiting the factorisability of a MAB–DCOP.

We randomly generated MAB–DCOP instances that can be encoded as acyclic factor graphs, with $n = 15$ and $m = 14$, and $|D_i| = 3$. Each utility function $U_j(\mathbf{x}_j)$ is governed by a set of normal distributions, one for each assignment to $\mathbf{x}_j$. In more detail, $U_j(\mathbf{x}_j) \sim \mathcal{N}(\mu(\mathbf{x}_j), \sigma^2(\mathbf{x}_j))$, where $\mu(\mathbf{x}_j)$ and $\sigma^2(\mathbf{x}_j)$ are uniformly drawn from intervals $[0, \mu_{\max}]$ and $[0, 1]$ respectively. Parameter $\mu_{\max}$ is used to control the relative importance that needs to be given to exploration and exploitation. When $\mu_{\max}$ is decreased, the received utilities become more noisy and the balance should be shifted towards exploration. Conversely, when $\mu_{\max}$ is increased, the optimal joint action becomes more easily identifiable, and agents start exploitation quite early on. For our experiments, we chose $\mu_{\max} = 1$ and $\mu_{\max} = 10$. These values were chosen during initial calibration to yield two sets of difficult problems, which simultaneously demonstrate the difference in required emphasis between exploration and exploitation.

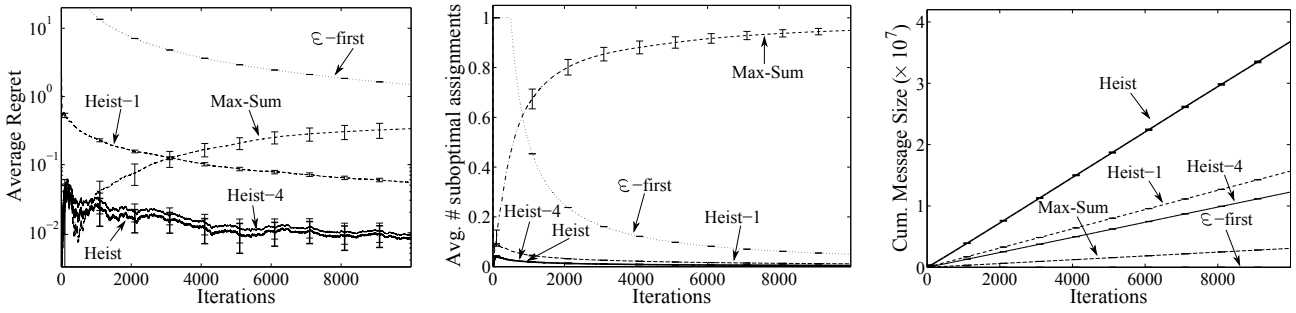
The results are shown in Figures 2 and 3 for $\mu_{\max} = 1$ and $\mu_{\max} = 10$ respectively. Each algorithm was run 64 times on both problem classes to obtain statistically significant results. Error bars indicate the standard error of the mean. Monolithic UCB is omitted from all figures, because its regret converged approximately a factor of 3^{15} slower than HEIST. This was expectated, as it regards the problem as a MAB with 3^{15} arms, instead of 14 MABs with 9 arms each.

Now, Figures 2(a) and 3(a) show the average regret for the remaining algorithms, while Figures 2(b) and 3(b) show the number of average suboptimal assignments. As can be observed from these figures, HEIST and HEIST-4 outperform all others in terms of regret (up to 1.5 orders of magnitude for $\mu_{\max} = 10$). We found that for $c \geq 8$, the per-



(a) Regret (b) Suboptimal joint assignments (c) Message size

Figure 2: Empirical results for $\mu_{\max} = 1$



(a) Regret (b) Suboptimal joint assignments (c) Message size

Figure 3: Empirical results for $\mu_{\max} = 10$

formance of HEIST (which was run for 30 iterations to ensure convergence) and HEIST-c coincide, indicating that the algorithm performs well even when conditions of Theorem 2 are not met. Max-sum—and indeed any algorithm that solves Equation 11—is clearly suboptimal, as its regret does not converge to zero, and it consistently produces suboptimal assignments. The fact the regret of max-sum *increases* is counter intuitive. However, additional experimentation showed that for smaller problem instances, the difference between HEIST and max-sum is much smaller, and their performance often coincides for problems with $m < 5$, while for $m = 50$, the difference in regret at $T = 10000$ was found to be more than 3 orders of magnitude. This leads us to believe that for larger problems, the non-linearity of Equation 9 is more pronounced, increasing the difference between the regret associated with assignments $X^+(t)$ and $X^*(t)$. The regret of the second DCOP based technique, ε-first, does converge to zero, but at a much slower pace than HEIST. Based on these results, we can conclude that both DCOP techniques are unsuitable for solving MAB–DCOPs.

Focusing on communication overhead, Figures 2(c) and 3(c) show the cumulative message size expressed as the number of floating point values exchanged between the agents. Compared to ε-first (which needs a negligible number of messages to coordinate once) and max-sum (which exchanges scalars, instead of sets of tuples), HEIST requires more communication. However, a good balance can be struck between solution quality and communication by reducing c to 4. Moreover, note that HEIST requires each agent to exchange only 600 values per iteration of the MAB–DCOP, a value that is well within the capabilities of bandwidth constrained embedded agents. Finally, the computation required by HEIST to solve problem instances with $n = 50$ and $T = 20000$ never exceeded 4 hours on a standard desktop PC, which is less than 200ms per agent per iteration. Again, this is well within the reach of embedded agents.

6. CONCLUSIONS AND FUTURE WORK

In this paper, we developed HEIST, a novel algorithm for coordination under stochasticity and the absence of *a priori* knowledge about the utility functions that govern the agents' interactions. HEIST solves MAB–DCOPs, an extension to the canonical DCOP framework, in which the local utility functions are transformed into MABs. By so doing, a MAB–DCOP becomes a sequential problem, instead of a single-shot optimisation problem, in which agents' need to trade off exploration and exploitation in a decentralised fashion. We formalised this trade off as a problem of maximising the UCB on the global utility, which we showed is a non-linear objective function. While previous algorithms have been shown to be incomplete, i.e. are not guaranteed to maximise such functions, HEIST is provably optimal. This is achieved by applying the GDL message passing algorithm on the GDL–UCB semi-ring, which is specifically designed to preserve the optimal joint variable assignment. We prove that the regret of HEIST is asymptotically optimal, i.e. it only differs from the optimal achievable regret by a constant factor. In addition, empirical results demonstrate that HEIST outperforms state of the art DCOP and MAB algorithms by up to 1.5 orders of magnitude.

For future work, we intend to further reduce the communication overhead of HEIST. In the empirical results, we already applied a technique for achieving this (HEIST-c), but this technique no longer carries the guarantee of optimality. Instead, we can let HEIST automatically calibrate the amount communication to suit the level of dynamism in the environment, while maintaining optimality.

Acknowledgements This work was part of the ORCHID project (`http://www.orchid.ac.uk/`).

7. REFERENCES

[1] S. M. Aji and R. J. McEliece. The Generalized Distributive Law. *IEEE Trans. Inf. Theory*, 46(2):325–343, 2000.

[2] J. Atlas and K. Decker. Coordination for uncertain outcomes using distributed neighbor exchange. *AAMAS'10*, pages 1047–1054, 2010.
[3] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite–time analysis of the multiarmed bandit problem. *Machine Learning*, 47:235–256, 2002.
[4] E. Even-Dar, S. Mannor, and Y. Mansour. Pac bounds for multi–armed bandit and markov decision processes. *COLT'02*, pages 255–270, 2002.
[5] S. Fitzpatrick and L. Meertens. Distributed coordination through anarchic optimization. In *Distributed Sensor Networks*, pages 257–295. Kluwer Academic Publishers, 2003.
[6] M. Jain, M. Taylor, M. Tambe, and M. Yokoo. DCOPs meet the real world: Exploring unknown reward matrices with applications to mobile sensor networks. *IJCAI'09*, pages 181–186, 2009.
[7] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory*, 47(2):498–519, 2001.
[8] T. L. Lai and H. Robbins. Asymptotically efficient adaptive allocation rules. *Adv. in Appl. Math.*, 6(1):4–22, 1985.
[9] T. Léauté and B. Faltings. E[DPOP]: Distributed constraint optimization under stochastic uncertainty using collaborative sampling. *IJCAI–09 DCR Workshop*, pages 87–101, 2009.
[10] T. Léauté and B. Faltings. Distributed constraint optimization under stochastic uncertainty. *AAAI'11*, pages 68–73, 2011.
[11] D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 2003.
[12] 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, 2005.
[13] 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.
[14] A. Petcu and B. Faltings. DPOP: A scalable method for multiagent constraint optimization. *IJCAI'05*, pages 266 – 271, 2005.
[15] A. Rogers, A. Farinelli, R. Stranders, and N. R. Jennings. Bounded approximate decentralised coordination via the max-sum algorithm. *Artif. Intell.*, 175(2), 2011.
[16] R. S. Sutton and A. G. Barto, editors. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
[17] M. Taylor, M. Jain, Y. Jin, M. Yokoo, and M. Tambe. When should there be a "Me" in "Team"?: Distributed multi-agent optimization under uncertainty. *AAMAS'10*, pages 109–116, 2010.
[18] J. Vermorel and M. Mohri. Multi-armed bandit algorithms and empirical evaluation. *ECML'05*, pages 437–448, 2005.

Appendix – Proofs

PROOF SKETCH OF THEOREM 2. We first show that operator $\max_{\succ}$ filters out tuples that cannot maximise the global UCB. In particular, if there is at least one incoming message, then $(\hat{\mu}_2, b_2^2)$ is dominated by $(\hat{\mu}_1, b_1^2)$ if and only if for any incoming tuple $(\hat{\mu}_3, b_3^2)$, $\hat{\mu}_1 + \hat{\mu}_3 + \sqrt{b_1 + b_3} > \hat{\mu}_2 + \hat{\mu}_3 + \sqrt{b_2 + b_3}$ holds. This implies that $\hat{\mu}_1 - \hat{\mu}_2 > \sqrt{b_2 + b_3} - \sqrt{b_1 + b_3}$. By definition, $b_3 = u_{\text{range}}\sqrt{\frac{2\ln t}{n_3(t)}}$ at time t, where $n_3(t)$ is the number of samples included in $\hat{\mu}_3$. Note that $1 \leq n_3(t) \leq t$, and can take any arbitrary value within this interval. That is, $u_{\text{range}}^2 \frac{2\ln t}{t} \leq b_3^2 \leq u_{\text{range}}^2 2\ln t$. This implies that if at least one among the second or the third clause in the definition does not hold, then tuple $(\hat{\mu}_2, b_2^2)$ cannot be discarded (i.e. it is not dominated by tuple $(\hat{\mu}_1, b_1^2)$. In addition, if there are no incoming messages, then breaking the first clause indicates that $(\hat{\mu}_2, b_2^2)$ cannot be discarded either. That is, $(\hat{\mu}_2, b_2^2)$ is dominated by $(\hat{\mu}_1, b_1^2)$ iff all the clauses hold.

The proof of the claim that the GDL message passing phase yields $Z_i(x_i)$, the maximum marginal UCB for each assignment, follows a similar argument to that of Theorem 3.1 in [1], and thus, is omitted for brevity. □

PROOF SKETCH OF THEOREM 3. At each time t, after GDL message passing has converged, the joint assignment with the maximum UCB is chosen (see Theorem 2). Suppose that each time t, HEIST chooses joint assignment $X^*(t) = \langle x_1^*(t), \ldots, x_n^*(t)\rangle$. In what follows, we estimate the expected number times $X^*(t) \neq X^{\mathbb{E}}$ is chosen, in order to estimate the regret of HEIST. In particular, let $N_T(X)$ denote the number of times HEIST chooses suboptimal joint assignment $X \neq X^{\mathbb{E}}$ before T. By estimating $N_T(X)$, we can estimate the number of times HEIST chooses a suboptimal joint assignment, and thus, derive a bound on its regret. That is,

$$R_{\text{HEIST}}(T) \leq \sum_{X \in \prod D_i} N_T(X) d(X) \tag{12}$$

We provide an upper bound for $\mathbb{E}[N_T(X)]$ as follows. Note that $\mathbb{E}[N_T(X)]$ can be estimated by the following sum:

$$\mathbb{E}[N_T(X)] \leq k + \sum_{t=1}^{T} P\left(X^*(t) = X, X \neq X^{\mathbb{E}}, N_{t-1}(X) \geq k\right) \tag{13}$$

The latter term can be further upper bounded by:

$$\sum_{t=1}^{T} P\left(\hat{\mu}(X(t),t) + b(X(t),t) \geq \hat{\mu}(X^{\mathbb{E}},t) + b(X^{\mathbb{E}},t), N_{t-1}(X) \geq k\right) \tag{14}$$

where $b(X,t) = \sqrt{2\ln(t)\sum_{j=1}^{m}\frac{(u_{\text{range}})^2}{n(\mathbf{x}_j,t)}}$. Intuitively, the probability that HEIST chooses $X^*(t) \neq X^{\mathbb{E}}$ can be bounded by the probability that $\hat{\mu}(X(t),t) + b(X(t),t) \geq \hat{\mu}(X^{\mathbb{E}},t) + b(X^{\mathbb{E}},t)$. This can be further bounded by:

$$\sum_{t=1}^{T}\sum_{s_j=k}^{t}\sum_{s=k}^{t} P\left(\hat{\mu}(X(s_j),s_j) + b(X(s_j),s_j) \geq \hat{\mu}(X^{\mathbb{E}},s) + b(X^{\mathbb{E}},s)\right) \tag{15}$$

Now, it is true that if $\hat{\mu}(X(s_j),s_j) + b(X(s_j),s_j) \geq \hat{\mu}(X^{\mathbb{E}},s) + b(X^{\mathbb{E}},s)$ holds then at least one of the following must hold:

1. $\hat{\mu}\left(X^{\mathbb{E}},s\right) + b\left(X^{\mathbb{E}},s\right) \leq \mu\left(X^{\mathbb{E}},s\right)$.
2. $\mu(X(s_j),s_j) \leq \hat{\mu}(X(s_j),s_j) + b(X(s_j),s_j)$.
3. $\mu\left(X^{\mathbb{E}},s\right) - \mu(X(s_j),s_j) \leq 2b(X(s_j),s_j)$.

This can be shown by using similar argument to that of Theorem 1 in [3], and thus, is omitted for brevity. By using McDiarmid's inequality, we can show that both (1) and (2) hold with probability t^{-4}, and if $k \geq \lceil\frac{u_{\text{range}}^2 8\ln T}{d(X)}\rceil$, then (3) does not hold. This implies that:

$$\mathbb{E}[N_T(\mathbf{x})] \leq k + \sum_{t=1}^{T}\sum_{s=1}^{t}\sum_{s_j=1}^{t} 2t^{-4} \leq k + \frac{\pi^2}{3} \tag{16}$$

for any $k \geq \lceil\frac{u_{\text{range}}^2 8\ln T}{d_{\mathbf{x}}}\rceil$. The last inequality is obtained from the Riemann Zeta Function for value of 2. Finally, substituting this into Equation 12 concludes the proof. □

PROOF SKETCH OF THEOREM 4. We can reduce all standard MAB problems to a MAB–DCOP with $m = 1$. According to [8], the best possible regret that an algorithm can achieve on standard MABs is $C\ln T$. Therefore, if there is an algorithm for MAB–DCOPs that provides better regret than $C\ln T$, then it also provides better regret bounds for standard MABs. □

Session 4B
Agent Societies

A Multiagent Evolutionary Framework based on Trust for Multiobjective Optimization

Siwei Jiang Jie Zhang Yew Soon Ong
School of Computer Engineering
Nanyang Technological University, Singapore
{sjiang1, zhangj, asysong}@ntu.edu.sg

ABSTRACT

In an Evolutionary Algorithm (EA) for optimization problems, candidate solutions to the problems are individuals in a population. They produce offsprings by taking evolutionary operators with user-specific control parameters. The challenge is then how to effectively select evolutionary operators and adjust control parameters from generation to generation and on different problems. We propose a novel multiagent evolutionary framework based on trust where each solution is represented as an intelligent agent, and evolutionary operators and control parameters are represented as services. Agents select services in each generation based on trust that measures the competency or suitability of the services for solving particular problems. Multiobjective Optimization Problems (MOPs) are used to showcase the value of our framework. Experimental studies on 35 benchmark MOPs show that our framework significantly improves the performance of the state-of-the-art EAs.

Categories and Subject Descriptors

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

General Terms

Design; Algorithms

Keywords

Evolutionary Algorithm; Multiagent Systems; Trust and Reputation; Multiobjective Optimization

1. INTRODUCTION

Evolutionary Algorithms (EAs) [1] are generic population-based stochastic search techniques inspired by biological evolution of nature selection for solving optimization problems. In EAs, candidate solutions to the problems play the role of individuals in a population. They produce offsprings by taking *evolutionary operators* (such as crossover and mutation) with user-specific *control parameters*. EAs are well known by its generality and simplicity that they often perform well approximating solutions to all types of problems in many fields such as engineering, economics, robotics, etc. However, evolutionary operators and control parameters may vary for different problems. It is time-consuming to determine the operators and parameters by the trial-and-error procedure. In addition, the competency of operators may vary with generations. For example, crossover is often powerful in the earlier stage of EAs, but mutation is effective when the solutions are similar with each other in the later stage of EAs. The challenge of EAs is thus how to effectively select evolutionary operators and adjust control parameters from generation to generation and on different problems.

Different techniques have been proposed to select evolutionary operators and control parameters in EAs. Igel et al. [3] propose CMA-ES as an evolution strategy that automatically adjusts control parameters, including the step-size, matrix mean and covariance. The methods of SaDE [5] and CoDE [8] not only adjust control parameters but also select evolutionary operators. However, both SaDE and CoDE introduce some other parameters whose values are predefined according to previous studies, which limits their generality.

In this paper, we propose a novel multiagent evolutionary framework based on trust where each solution is represented as an intelligent agent, and the pairs of evolutionary operators and control parameters are represented as services. In our framework, the agents model the trustworthiness of the services, based on whether the agents' offsprings produced by using the services survive to the next generations, which represents the dynamic competency or suitability of the services from generation to generation and on particular optimization problems. The agents will then select the services with the probabilities correlated to the trustworthiness of the services. To demonstrate the value of our framework, we consider the challenges of Multiobjective Optimization Problems (MOPs) as a case study, whereas it is generally applicable to other optimization problems. Experimental results on 35 benchmark MOPs confirm that our framework significantly improves the performance of the state-of-the-art EAs. The present work thus represents a promising step towards the use of multiagent based paradigms in the design of novel EAs as composing of intelligent agents that adopt trust modeling techniques for selecting evolutionary operators and control parameters in multiagent-based EAs.

2. RELATED WORK

CMA-ES [3] is a representative evolution strategy (ES) to adaptively adjust control parameters for solving MOPs. In CMA-ES, only one type of evolutionary operator is used to produce offsprings by a Gaussian mutation. The mean

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

and variance of the Gaussian distribution are adjusted with an iteration procedure. Our framework not only adjusts control parameters but also selects evolutionary operators. SaDE [5] and CoDE [8] are the two representative algorithms proposed to select evolutionary operators and adjust control parameters for solving single objective optimization problems. In SaDE, operators and parameters are gradually self-adapted by learning from their previous experience in generating promising solutions. In each generation, operators and parameters are assigned to different individuals in the current population according to the selection probabilities learned from the previous generations. However, SaDE computes simple statistics on the experience only after each 50 generations. In our framework, the competency or suitability of evolutionary operators and control parameters (referred to as the trustworthiness of services) is modeled by cumulating all previous experience based on well established probabilistic modeling and in a dynamic manner. Also, SaDE adjusts control parameters based on a normal distribution with a predefined mean based on the authors' prior knowledge. The CoDE algorithm randomly combines three DE operators and three predefined parameters to generate offsprings. Although CoDE obtains better performance over SaDE, its setting of the control parameters relies on some prior knowledge. All in all, SaDE and CoDE both introduce some other parameters whose values are predefined based on previous studies on single objective optimization problems, which limits their generality to other problems, e.g. more complex Multiobjective Optimization Problems (MOPs). In contrast, our selection of operators and control parameters does not rely on any prior knowledge about the problems. In addition, we design our framework as a multiagent system where candidate solutions are represented as intelligent agents capable of learning, cooperation and adaptation. This design offers great flexibility and extendability for EAs to employ advanced multiagent technologies for solving complex optimization problems.

Multiagent technologies have recently been widely used to design Evolutionary Algorithms (EAs) for solving complex problems [6]. For example, Stonedahl et al. [7] propose a distributed multiagent-based Genetic Algorithm (GA) to study how the network density of connections and the interactions between agents affect the performance of the GA. In the work of Zhong et al. [11] for solving single objective optimization problems, every solution is considered as an agent and all agents live in a lattice-like environment. The actions of agents are advanced evolutionary operator (such as orthogonal crossover and self-learning operators), but the agents are not autonomous because they select actions only based on predefined probabilities. In our framework, agents autonomously select services by learning the trustworthiness of the services. Trust plays a crucial role in agent-based service selection [10, 9]. It is used by agents to measure the quality of services and select services of high quality. One particularly effective way of modeling trust is to use the collective opinions of all agents about the services. We adopt this method in our framework.

Thus, the contributions of our current work can be summarized as follows: 1) majority of the adaptive EAs have been proposed to work with single objective optimization problems. Our generic framework can also be adopted to solve MOPs and other complex optimization problems; 2) the few existing adaptive EAs for MOPs adjust only control parameters, whereas agents in our framework can also select evolutionary operators; 3) to the best of our knowledge, multiagent technologies have been adopted to design adaptive EAs for solving MOPs for the first time; 4) our framework is also the first attempt to consider the use of trust modeling for measuring the dynamic competency of evolutionary operators and control parameters.

3. BACKGROUND ON MOEA

We demonstrate our framework on solving Multiobjective Optimization Problems (MOPs) [1]. MOPs involve several conflicting objectives to be optimized simultaneously. A minimization of MOPs can be stated as follows:

$$\begin{array}{rcl} min & \mathcal{F}(\vec{x}) & = (f_1(\vec{x}), \ldots, f_m(\vec{x})) \\ s.t. & g(\vec{x}) & \leq 0,\ h(\vec{x}) = 0,\ \vec{x} \in \Omega \end{array} \quad (1)$$

where $\vec{x} = (x_1, \ldots, x_D)$, Ω is decision (*variable*) *space*, R^m is *objective space*, and $\mathcal{F} : \Omega \rightarrow R^m$ consists of m real-valued objective functions with constraints $g(\vec{x}) \leq 0, h(\vec{x}) = 0$, and the feasible solution space is $\Omega = \Pi_{i=1}^{D}[LB_i, UB_i]$.

The challenge of MOPs is to find a *Pareto set* (PS) including non-dominated solutions which are evenly scattered along *Pareto front* (PF). Multiobjective Evolutionary Algorithms (MOEAs) have been well established as efficient approaches to solve various MOPs [1].

In MOEAs, the first population of solutions is randomly generated as $X_g = \{\vec{x}_{i,g} | i = 1, \ldots, NP, g = 0\}$, where NP is the population size and g is the generation index. The next population is produced by evolutionary operators. We take the "DE/rand/1/bin" operator as an example. At first, the operator generates a vector $\vec{v}_{i,g}$ base on population X_g.

$$\vec{v}_{i,g} = \vec{x}_{r1,g} + F \cdot (\vec{x}_{r2,g} - \vec{x}_{r3,g}) \quad (2)$$

where $r1, r2, r3 \in [1, NP]$ are random integer numbers and $r1 \neq r2 \neq r3 \neq i$. The control parameter F is the scaling factor which amplifies or shrinks the difference vectors.

After that, "DE/rand/1/bin" applies the binomial crossover operation to produce the offspring vectors:.

$$\begin{array}{c} U_g = \{\vec{u}_{i,j,g} | i = 1, \ldots, NP, j = 1, \ldots, D\} \\ \vec{u}_{i,j,g} = \begin{cases} \vec{v}_{i,j,g} & \text{if } rand_j(0,1) \leq CR \text{ or } j = j_{rand} \\ \vec{x}_{i,j,g} & \text{otherwise.} \end{cases} \end{array} \quad (3)$$

where $rand_j(0,1) \in [0,1]$ is a uniformly distributed random number, $j_{rand} \in [1, D]$ is a randomly chosen integer. If $\vec{u}_{i,j,g} < LB_j$, it is set to LB_j, if $\vec{u}_{i,j,g} > UB_j$, set to UB_j. The control parameter CR is the probability for crossover.

Then, MOEAs select part of offsprings to enter the next generation ($\vec{u}_{i,g} \rightarrow X_{g+1}$). MOEAs can be generally categorized into two major classes: decomposition-based (called MOEA/D) [4] and Pareto dominance-based MOEAs [2, 12].

- In MOEA/D, $\vec{u}_{i,g} \rightarrow X_{g+1}$ if $\vec{u}_{i,g} \succeq \vec{x}_{j,g+1}$ $(\forall \vec{x}_{j,g+1} \in X_{g+1})$[1] under, for example, Tchebycheff approach [4].
- In Pareto dominance-based MOEAs, $\vec{u}_{i,g} \rightarrow X_{g+1}$ if $\vec{u}_{i,g} \succeq \vec{x}_{j,g+1}$ $(\forall \vec{x}_{j,g+1} \in X_{g+1})$ under, for example, crowding distance (NSGAII) [2] or neighborhood density estimator (SPEA2) [12].

The performance of MOEAs is determined by the operators and their parameters (i.e. the operator "DE/rand/1/bin", and parameters F and CR in the operator mentioned above). The purpose of our framework is to select proper evolutionary operators and control parameters in EAs (i.e. MOEAs).

[1]"$\succeq$" means "be better than or equal".

4. OUR FRAMEWORK

In MOEAs, solutions in each generation produce offsprings by performing evolutionary operators with some control parameters. A plenty of effective evolutionary operators have been proposed, such as "DE/rand/1/bin", "DE/rand/2/bin", "DE/current-to-rand/1/bin" [8], Simulated Binary Crossover (SBX), and Polynomial mutation [2]. These operators, configured with different control parameters, exhibit distinguishing competence on different MOPs. The offsprings produced by some operators and parameters may be able to survive to the next generation, but some offsprings cannot.

In our multiagent evolutionary framework, each solution is represented as an agent. The pairs of evolutionary operators with corresponding control parameters are represented as services. In each generation, an agent selects a service to produce a new offspring agent (i.e., by Equations 2 and 3), which is also a solution. The new offspring agent competes with other agents in the environment. If the offspring agent can survive to the next generation, it means that the service provides a positive outcome, otherwise, the service provides a negative outcome. The trustworthiness of services can be used to represent the competency of the services in producing positive outcomes. The larger number of outcomes a service can produce, the more suitable the service is to solve the given problem. Thus, agents in our framework model the trustworthiness of the services based on the number of positive and negative outcomes provided by the services in the past generations. The modeling results will be used by the agents to make decisions on which services to consume.

4.1 Probabilistic Modeling of Trustworthiness

The trustworthiness of services is normally modeled based on the number of positive and negative outcomes produced by them in the past. If we define s as the number of positive outcomes and f as the number of negative outcomes provided by a service S, formulated as follows:

$$\begin{cases} s = s + 1 & \text{if } \vec{u}_{i,g} \rightarrow X_{g+1} \\ f = f + 1 & \text{otherwise} \end{cases} \tag{4}$$

where $\vec{u}_{i,g} \rightarrow X_{g+1}$ means that the offspring $\vec{u}_{i,g}$ produced in the generation g by the service can survive to the next generation $g+1$. Whether $\vec{u}_{i,g} \rightarrow X_{g+1}$ is determined based on different methods in MOEAs (see Section 3).

Beta distribution is commonly used to model the distribution of a random variable representing the unknown probability of a binary event. The Beta probability density functions (PDF) of service S can then be formulated as:

$$\text{Beta}(p(S)|\alpha,\beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} p(S)^{\alpha-1}(1-p(S))^{\beta-1} \tag{5}$$

where $0 \leq p(S) \leq 1$ and $\alpha, \beta > 0$ with the restriction that $p(S) \neq 0$ if $\alpha < 1$ and $p(S) \neq 1$ if $\beta < 1$.

The trustworthiness of S is then the probability expectation value of the Beta distribution, which represents the relative frequency of positive outcomes in future events [10].

$$T(S) = \frac{\alpha}{\alpha+\beta}, \quad \text{where} \quad \alpha = s+1, \ \beta = f+1 \tag{6}$$

4.2 Trustworthiness of Service

In this paper, we model the trustworthiness of services by adopting probabilistic modeling introduced in the previous section. One thing to note here is that MOEAs generally involve much randomness. A evolutionary operator configured with the same control parameters may still generate different offsprings because of the random values of $r1$, $r2$, $r3$ and $rand_j(0,1)$ in Equations 2 and 3. Due to this randomness, the trustworthiness of a service cannot be accurately estimated by a small number of outcomes produced by the service for a particular family of agents (a solution and its offsprings). Instead, in our framework, it is modeled based on the outcomes produced by the service for all agents in the past generations, which is referred to as reputation [10].

A service is represented by a evolutionary operator and some control parameters. The evolutionary operator can be any operator from a list of operators $O = \{O_1, O_2, \ldots, O_{|O|}\}$ proposed in MOEAs, where $|O|$ is the number of available evolutionary operators. Given a specific operator $O_k \in O$ in the service, there will be a set of control parameters $C^k = \{C^k_l | l = 1, \ldots, |C^k|\}$ associated with the operator O_k, where $|C^k|$ is the number of control parameters. For example, the operator "DE/rand/1/bin" has two control parameters (CR and F) associated with it (see Section 3). Assume that a parameter C^k_l takes a continuous value in the range as $C^k_l \in [0,1]$. In order to effectively learn the performance of a control parameter, we divide the range $[0,1]$ into a set of q disjoint segments as $L = \{[0, \frac{1}{q}), [\frac{1}{q}, \frac{2}{q}), \cdots, [\frac{q-1}{q}, 1]\}$. Thus, a service can be formally defined as a tuple (O_k, C^k) where $C^k = \{C^k_l | C^k_l = L(C^k_l), l = 1, \ldots, |C^k|\}$ and $L(C^k_l)$ is one of the segments in L for the parameter C^k_l. In another word, a service is a tuple of a evolutionary operator and a set of segments for corresponding control parameters. Here, we do not distinguish a parameter from its segment for simplicity.

For the service (O_k, C^k), we first compute the trustworthiness of the operator O_k. It is modeled based on the number of positive and negative outcomes generated by the agents performing this operator in the past generations. The total number of positive and negative outcomes up to the current generation g is aggregated as follows:

$$\begin{cases} s_g(O_k) &= (1-\eta) \cdot s_{g-1}(O_k) + \eta \cdot N_{g,s}(O_k) \\ f_g(O_k) &= (1-\eta) \cdot f_{g-1}(O_k) + \eta \cdot N_{g,f}(O_k) \end{cases} \tag{7}$$

where $N_{g,s}(O_k)$ and $N_{g,f}(O_k)$ are the number of positive and negative outcomes produced by the agents performing the operator O_k in the current generation g, respectively. The parameter $0 \leq \eta \leq 1$ is to determine how much to consider the current and historical information, where $\eta = 0$ means that only the historical information is considered, whereas $\eta = 1$ only the current information is utilized. After having $s_g(O_k)$ and $f_g(O_k)$, the trustworthiness of the operator O_k in the current generation g, $T_g(O_k)$, can then be computed according to Equation 6.

In general, one operator is suitable for some specific types of problems, but may not work well for other types. Even for the same problem, the competency of the operator may vary in different generations. For example, the operator "DE/ran/1/bin" is suitable to multi-modal problems, which has slow convergency in the earlier stage but exhibits strong exploration in the later stage of EAs. Based on this phenomenon, the trustworthiness of the operator needs to reflect the varying competency of the operator under the condition where trust is hard to build up, but easy to lose.

The aggregation function in Equation 7 is then revised as:

$$\begin{cases} s_g &= (1-T_{g-1}) \cdot s_{g-1} + T_{g-1} \cdot N_{g,s} \\ f_g &= (1-T_{g-1}) \cdot f_{g-1} + T_{g-1} \cdot N_{g,f} \end{cases} \tag{8}$$

where O_k is dropped out for clarity and T_{g-1} is the trust-

worthiness of operator O_k in generation $g-1$. Equation 8 has two important advantages. It does not have predefined parameters, compared to Equation 7 that has the parameter η. Equation 8 also satisfies the above mentioned condition. When the trustworthiness of the operator in the last generation $g-1$, $T_{g-1}(O_k)$ is low, the operator needs more positive outcomes $N_{g,s}(O_k)$ to build up its trust in the current generation g. When $T_{g-1}(O_k)$ is high, $1-T_{g-1}(O_k)$ is low, meaning that the less consideration will be given to historical information. The trustworthiness of the operator $T_g(O_k)$ will be easy to decline when the number of negative outcomes in the current generation $N_{g,f}(O_k)$ is large.

For the service (O_k, C^k), we then compute the trustworthiness of each parameter in C^k (i.e. the value range segment corresponding to each parameter). When computing the trustworthiness a parameter, we also need to consider the operator the parameter is associated with. Take the parameter C_l^k as an example. The trustworthiness of C_l^k associated with O_k in the current generation g, denoted as $T_g(C_l^k|O_k)$, can be calculated in the similar way as calculating the trustworthiness of the operator O_k (Equation 8), by counting the numbers of positive and negative outcomes produced by the operator O_k with the parameter C_l^k, which are $N_{g,s}(C_l^k|O_k)$ and $N_{g,f}(C_l^k|O_k)$ respectively.

After having the trustworthiness of the evolutionary operator O_k, which is $T_g(O_k)$, and each control parameter C_l^k given O_k, which is $T_g(C_l^k|O_k)$, we can then compute the trustworthiness of the service (O_k, C^k) by assuming the control parameters are independent, as follows:

$$T_g(O_k, C^k) = T_g(O_k) \cdot \prod_{l=1}^{|C^k|} T_g(C_l^k|O_k) \tag{9}$$

4.3 Trust-based Service Selection

In our framework, agents select services based on the computed trust results of the services. In order to balance between exploitation and exploration, services are selected in a probabilistic manner where the probability for a service to be selected is proportional to its trust. More formally, there are $\sum_k^{|O|} |C^k| \cdot m$ services in total because there are $|O|$ evolutionary operators, each operator O_k is associated with $|C^k|$ control parameters, and each parameter is represented by one of the q value range segments. The probability for service (O_k, C^k) with the trust $T_g(O_k, C^k)$ in the current generation g to be selected in the next generation $g+1$ is:

$$p(O_k, C^k) = \frac{T_g(O_k, C^k)}{\sum_k^{\sum_k^{|O|} |C^k| \cdot m} T_g(O_k, C^k)} \tag{10}$$

Note that after an agent selects a service, e.g. (O_k, C^k), each control parameter in C^k, e.g. C_l^k, is a value range segment in L, not a specific value. In order for the service to be used by the agent to produce an offspring, a specific value for the parameter C_l^k is needed. We assume that the values of the parameter C_l^k follow a normal distribution in the range of $L(C_l^k)$ as Normal$(\mu_g(C_l^k), \sigma)$ where $\mu_g(C_l^k)$ and $\sigma = \frac{1}{3q}$ are the mean and standard deviation, respectively, and $C_l^k \in [0, 1]$. The mean $\mu_g(C_l^k)$ is calculated as follows:

$$\begin{aligned} \mu_g(C_l^k) \;=\; & (1 - T_{g-1}(C_l^k|O_k)) \cdot \mu_{g-1}(C_l^k) \\ & + T_{g-1}(C_l^k|O_k) \cdot \text{Mean}(V_g(C_l^k|O_k)) \end{aligned} \tag{11}$$

where $V_g(C_l^k|O_k)$ is the set of the values of the parameter C_l^k, which produces positive outcomes for the agent performing the operator O_k in the current generation g. Mean$(V_g(C_l^k|O_k))$ is the mean of the values in $V_g(C_l^k|O_k)$. The rationale behind Equation 11 is that the effectiveness of the parameter C_l^k measured by $T_{g-1}(C_l^k|O_k)$, reflects the appropriation of its mean $\mu_g(C_l^k)$ up to the generation $g-1$. To cope with the dynamics of the effectiveness of $\mu_g(C_l^k)$, we formulate it in a similar spirit as Equation 8.

5. EXPERIMENTATION

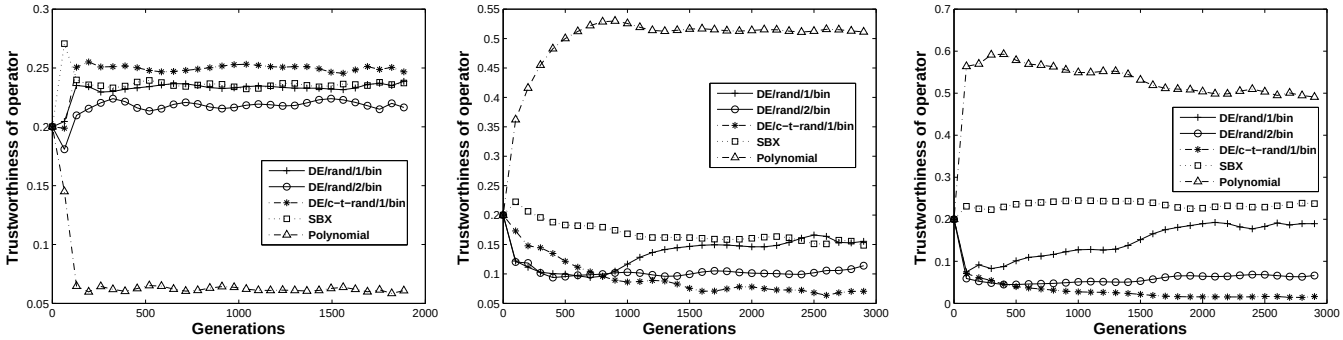
The experiments are carried out on jMetal 3.1[2], a Java-based framework aimed at facilitating the development of metaheuristics for solving MOPs. The benchmark problems include 35 test instances: 5 MOPs in the ZDTx family problems (ZDT1-4 and ZDT6 with 2 objectives), 7 MOPs in the DTLZx family problems (DTLZ1-7 with 3 objectives), and 23 MOPs in the CEC2009 MOEA competition. Among the problems used in the CEC2009 MOEA competition that involves unconstrained functions, UF1-7 have 2 objectives, UF8-10 3 objectives, and UF11-13 5 objectives. In addition, The problems CF1-10 have one constraint except CF6-7 have two constraints. The *decision variables* in the Pareto sets (PSs) of the ZDTx and DTLZx are independent, and those in the CEC2009 MOEA competition are dependent. The 35 MOPs have different geometrical shapes in *objective space* such as concave, convex, linear, discrete, uni-modal and multi-modal Pareto fronts (PFs).

The experimental settings are outlined as follows. The number of decision variables D used in ZDT1-3 is 30, $D = 10$ in ZDT4 and ZDT6, $D = 7$ in DTLZ1, $D = 12$ in DTLZ2-6, $D = 22$ in DTLZ7, $D = 30$ in UF1-13, and $D = 10$ in CF1-10. In MOEA/D, the population size NP is decided by the number of weight vectors C_{H+m-1}^{m-1} (m is the number of objectives, H is a predefined integer). For problems with two objectives, $NP = 100$ by setting $H = 99$, $NP = 153$ for tri-objective problems ($H = 16$), $NP = 715$ for five-objective problems ($H = 9$). The other algorithms have the same population size as MOEA/D on different MOPs. We set the maximum number of function evaluations (FEs) to be $300,000$ and independent run times to be 30.

We compare with classic MOEAs (NSGAII [2], SPEA2 [12] and MOEA/D [4]). We also compare with the other approaches (CMA-ES [3], SaDE [5] and CoDE [8]) that select evolutionary operators and/or control parameters. Five evolutionary operators are considered, including "DE/rand/1/bin", "DE/rand/2/bin", "DE/current-to-rand/1/bin" [8], "SBX", and "Polynomial mutation" [2]. In NSGAII and SPEA2, the control parameters for SBX are set to $\eta_c = 20$ and $p_c = 0.9$, and those for Polynomial mutation are set to $\eta_m = 20$ and $p_m = 1/D$. In MOEA/D, the control parameters of the operator "DE/rand/1/bin" are set to $CR = 1.0$ and $F = 0.5$. The update approach used in decomposition-based MOEAs is the Tchebycheff. In our framework, the number of value range segments for the control parameters is set to $q = 3$.

All the algorithms are evaluated by the hypervolume metric, which is strictly monotonic with regard to Pareto dominance [12]. The obtained results are compared using median values and interquartile range (IQR). In order to have statistically sound conclusions, the Wilcoxon rank sum test with 95% confidence level is conducted on the experiment results.

[2]http://jmetal.sourceforge.net



(a) DTLZ1 by NSGAII-T (b) UF1 by NSGAII-T (c) CF1 by NSGAII-T

Figure 1: The Trustworthiness of Operators $T_g(O)$ Derived by NSGAII-T on DTLZ1, UF1 and CF1

5.1 Improving MOEAs by Our Framework

In this experiment, we extend the classic MOEAs (NSGAII [2], SPEA2 [12] and MOEA/D [4]) by our framework that selects the five evolutionary operators and adjusts control parameters based on trust. The extended versions of NSGAII, SPEA2 and MOEA/D are NSGAII-T, SPEA2-T and MOEA/D-T, respectively. The purpose is to evaluate whether they can be improved by our framework.

Table 1: Statistical Comparison Results of Classic MOEAs versus Those Extended by Our Framework

	NSGAII-T	SPEA2	SPEA2-T	MOEA/D	MOEA/D-T
NSGAII	18/13/4	17/11/7	19/8/8	19/7/9	22/5/8
NSGAII-T		14/8/13	15/8/12	17/6/12	20/5/10
SPEA2			19/12/4	14/6/15	17/8/10
SPEA2-T				14/7/14	17/6/12
MOEA/D					22/10/3

Table 3 (in the end of the paper) shows the detailed experimental results, where each tuple reports the median and IQR of hypervolume over 30 independently runs on 35 MOPs with 300,000 FES. Table 1 shows the win/tie/lose ($w/t/l$) statistical results under the Wilcoxon rank sum test with 95% confidence level. Each tuple $w/t/l$ means that the algorithm at the corresponding column wins on w MOPs, ties on t MOPs, and loses on l MOPs, compared to the algorithm at the corresponding row. The results show that the $w/t/l$ values between the extended versions by our framework and the classic MOEAs and are 18/13/4, 19/12/4, 22/10/3, respectively. This indicates that our framework can significantly improve the performance of the classic MOEAs. We also see that MOEA/D-T is the most effective to solve MOPs than the classic MOEAs and the other extended MOEAs (NSGAII-T and SPEA2-T).

5.2 Comparison with Adaptive Approaches

In this experiment, we implement the other adaptive approaches for selecting evolutionary operators and/or control parameters (CMA-ES [3], SaDE [5], and CoDE [8]) to extend MOEA/D for solving MOPs. The purpose is to compare the effectiveness of them with our framework.

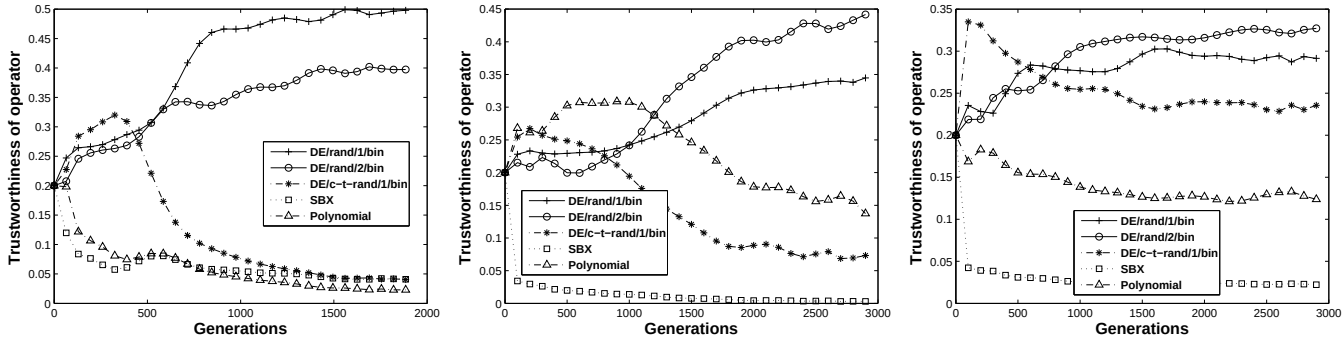
CMA-ES uses only one evolutionary operator (Gaussian mutation), and adjusts the mean and variance of Gaussian distribution in variable space. SaDE and CoDE select operators among "DE/rand/1/bin", "DE/rand/2/bin" and "DE/current-to-rand/1/bin". In SaDE, the control parameters are generated by normal distribution, where $\sigma_{CR} = 0.3$, $\mu_F = 0.5$ and $\sigma_F = 0.1$. SaDE introduces four predefined parameters, including the learning period of 50 generations. CoDE combines the three operators with a set of fixed parameter settings, including $[CR = 0.1, F = 0.1]$, $[CR = 1.0, F = 0.5]$ and $[CR = 0.2, F = 0.8]$. Our framework selects operators among "DE/rand/1/bin", "DE/rand/2/bin", "DE/current-to-rand/1/bin", "SBX" and "Polynomial mutation". To have a fair comparison, we implement MOEA/D-T3 that uses our framework to select among only the first three operators, which is the same as SaDE and CoDE. The algorithm MOEA/D-T' is implemented to use another version of our framework that models the trustworthiness of services with a fixed parameter $\eta = 0.3$ using Equation 7.

Table 4 (in the end of the paper) shows the detailed experimental results of comparing CMA-ES, SaDE and CoDE with MOEA/D-T3, MOEA/D-T' and MOEA/D-T, where each tuple reports the median and IQR of hypervolume over 30 independently runs on 35 MOPs with 300,000 FES. Table 2 shows the win/tie/lose ($w/t/l$) statistical comparison results of the six algorithms under the Wilcoxon rank sum test with 95% confidence level.

Table 2: Comparison Results for CMA-ES, SaDE, CoDE, MOEA/D-T3, MOEA/D-T', MOEA/D-T

	SaDE	CoDE	MOEA/D-T3	MOEA/D-T'	MOEA/D-T
CMA-ES	27/4/4	25/8/2	27/7/1	26/8/1	30/4/1
SaDE		10/17/8	17/13/5	14/17/4	21/8/6
CoDE			11/21/3	10/16/9	22/11/2
MOEA/D-T3				6/20/9	9/24/2
MOEA/D-T'					14/16/5

The $w/t/l$ values between MOEA/D-T3 and CMA-ES is 27/7/1, indicating that selecting evolutionary operators is beneficial for solving MOPs. The $w/t/l$ values between MOEA/D-T3 and SaDE is 17/13/5. Our MOEA/D-T3 not only has less predefined parameters than SaDE, but also is more effective than SaDE. The $w/t/l$ value between MOEA/D-T3 and CoDE is 11/21/3. MOEA/D-T3 is more effective than CoDE. CoDE fixes the two control parameters CR and F before the algorithm starts. MOEA/D-T3 learns the parameters as the algorithm progresses. It demonstrates that parameter learning based on trust in our framework is able to automatically adjust control parameters on different MOPs. The $w/t/l$ values between MOEA/D-T and CMA-ES, SaDE, CoDE, MOEA/D-T3, MOEA/D-T' are 30/4/1, 21/8/6, 22/11/2, 9/24/2 and 14/16/5, respectively. Our



(a) DTLZ1 by MOEA/D-T (b) UF1 by MOEA/D-T (c) CF1 by MOEA/D-T

Figure 2: The Trustworthiness of Operators $T_g(O)$ Derived by MOEA/D-T on DTLZ1, UF1 and CF1

MOEA/D-T is the best among the six algorithms. The selection on the two extra evolutionary operators "SBX" and "Polynomial mutation" does show advantages. MOEA/D-T outperforms MOEA/D-T', confirming that the dynamic modeling of the trustworthiness of services is more effective.

5.3 Effect of PSs on Operator Selection

An important factor that affects the searching ability of Pareto dominance-based MOEAs (e.g. NSGAII) in variable space is the interdependency among the decision variables in Pareto sets (PSs). In this experiment, we want to examine the effect of PSs on the selection of evolutionary operators. We evaluate NSGAII-T on the problems DTLZ1, UF1 and CF1. In DTLZ1, decision variables in PSs are independent, but in UF1 and CF1, the decision variables are interdependent. Figure 1 shows the trustworthiness of operators $T_g(O)$ in different generations. We can see that in Pareto dominance-based MOEAs, the evolutionary operators SBX and Polynomial mutation show great contributions. It is also evident that SBX does better in independent PSs (DTLZ1) than interdependent PSs (UF1 and CF1), whereas Polynomial mutation is more suitable to deal with the nonlinear variable dependencies than independent variables.

5.4 Trustworthiness of Evolutionary Operators

The operators "DE/rand/1/bin" and "DE/rand/2/bin" are quite similar. However, "DE/current-to-rand/1/bin" is different and it is formulated as follows:

$$\vec{v}_{i,g} = \vec{x}_{i,g} + rand() \cdot (\vec{x}_{r1,g} - \vec{x}_{i,g}) + F \cdot (\vec{x}_{r2,g} - \vec{x}_{r3,g}) \quad (12)$$

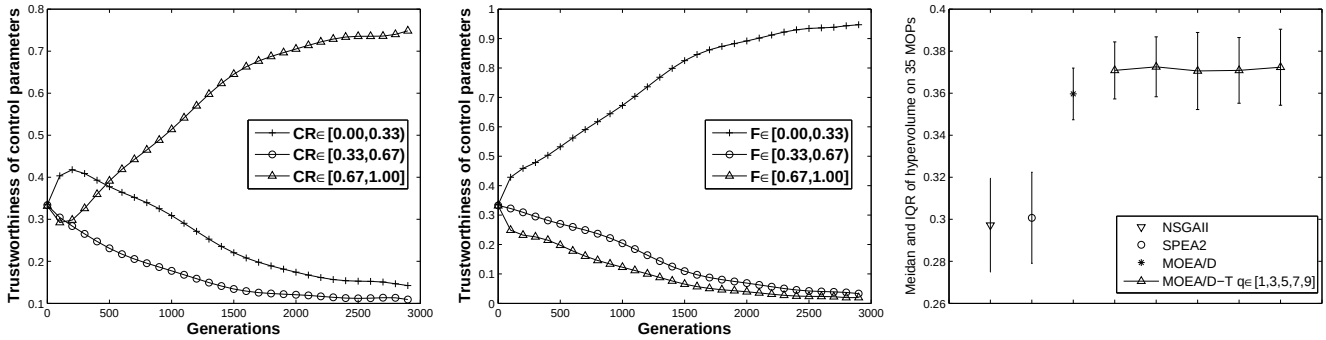
where $r1, r2, r3 \in [1, NP]$ are random integer numbers and $r1 \neq r2 \neq r3 \neq i$, $\vec{x}_i$ is the base vector, and $rand() \in (0, 1)$ is a uniform random value. Operator "DE/current-to-rand/1/bin" generates the new offsprings based on vector $\vec{x}_i$, whereas "DE/rand/1/bin" and "DE/rand/2/bin" search new agent in the global region.

The effectiveness (competency) of evolutionary operators is evaluated by the trustworthiness of them in our MOEA/D-T. In this experiment, we investigate the different competency of the operators in different generations on different MOPs. Figure 2 shows the trustworthiness of operators $T_g(O)$ on DTLZ1, UF1, and CF1 by MOEA/D-T in different generations. All results are means of 30 independent runs. We can see that the trustworthiness (competency) of the operators vary from generations to generations and on different problems. Under the decomposition-based MOEA method, "DE/rand/1/bin", "DE/rand/2/bin" and "DE/current-to-rand/1/bin" are more effective than "SBX" and "Polynomial". The trustworthiness of "DE/current-to-rand/1/bin" increases in the earlier stage then gradually decreases in the later stage, whereas the trustworthiness of "DE/rand/1/bin" and "DE/rand/2/bin" gradually increases as MOEA/D-T progresses. "DE/current-to-rand/1/bin" has the search bias based on the base vector ($\vec{x}_i$) and larger perturbation ($rand()$). In the earlier stage of MOEAs, it has good performance due to the biased search. But its performance gradually deteriorates in the later stage because of the uncertain perturbation. "DE/rand/1/bin" and "DE/rand/2/bin" do not prefer any search direction but they have strong exploration capability. The competency of them is low in the earlier stage because of their unbiased search. But in the later stage, they are more effective than "DE/current-to-rand/1/bin" due to the better exploration. Thus, the trustworthiness of the operators modeled by our framework well reflects their true competency.

5.5 Trustworthiness of Control Parameters

The effectiveness (suitability) of control parameters is also evaluated by the trustworthiness of them in MOEA/D-T. In this experiment, we investigate the different suitability of the control parameters in different generations. Figures 3(a) and 3(b) show the trustworthiness of the control parameters (different value range segments for CR and F respectively), $T_g(CR)$ and $T_g(F)$, on the problem UF1 by MOEA/D-T, where $\{CR, F \in [0, 1]\}$ is divided into three segments. Because of space limitation, we only show the results of the parameters for the operator "DE/rand/2/bin".

The trustworthiness of $CR \in [0.67, 1.00]$ is high in the later stage on UF1. The larger value of CR makes the operator to search in a broad region, and it is beneficial for MOEAs to maintain the population diversity. The trustworthiness of $F \in [0.00, 0.33)$ gradually increases on UF1. As the algorithm progresses, the agents (represent the solutions) spread more evenly. It means that the difference between agents (i.e., $\vec{x}_{r1,g} - \vec{x}_{r_2,g}$ in Equation 2) becomes larger. So, in the later stage, the operator "DE/rand/2/bin" needs to adjust the parameter F to be small for exploitation to search in a neighboring region. Thus, the trustworthiness of the control parameters modeled well reflects the varying competency of them in different generations.



(a) UF1 by MOEA/D-T (b) UF1 by MOEA/D-T (c) Effect of Control Parameters

Figure 3: (a, b) Trustworthiness of Parameters $T_g(CR|O)$ and $T_g(F|O)$ Derived by MOEA/D-T on UF1 and O = "DE/rand/2/bin" in Different Generations; (c) MOEA/D-T with Different Values of Parameter q

5.6 The Effect of Parameter q

Our framework has only one predefined parameter q, which is the number of value segments for control parameters. To investigate the impact of this parameter setting, MOEA/D-T with $q = \{1, 3, 5, 7, 9\}$ are tested on 35 MOPs. Figure 3(c) shows the median and IQR of hypervolume derived from NSGAII, SPEA2, MOEA/D and MOEA/D-T over 30 independent runs. It is evident that MOEA/D-T is not sensitive to the setting of q. For all q values, MOEA/D-T outperforms NSGAII, SPEA2 and MOEA/D.

6. CONCLUSION AND FUTURE WORK

In this paper, a novel multiagent evolutionary framework based on trust is proposed to effectively select evolutionary operators and adjust control parameters (represented as services), for solving complex optimization problems (such as MOPs). In the framework, agents (representing solutions) automatically select services by modeling their trustworthiness based on the number of offsprings produced using them will survive to the next generation. Experiments carried out on 35 benchmark MOPs confirm that our framework significantly improves the performance of the classic MOEAs (NSGAII, SPEA2 and MOEA/D) and outperforms the other three adaptive approaches (CMA-ES, SaDE and CoDE).

For future work, we will examine our framework in a distributed multiagent system where only partial (local and neighboring) information about the outcomes of services is known to agents, towards the development of a distributed framework. We will also investigate the performance of our framework on other complex problems, such as constraint optimization, expensive optimization problems, etc.

Acknowledgment

This work is partially supported by the Media Development Authority of Singapore, Singapore-MIT GAMBIT Game Lab and the Center for Computational Intelligence (C2I) at the Nanyang Technological University, Singapore.

7. REFERENCES

[1] C. A. C. Coello. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36, 2006.

[2] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transaction on Evolutionary Computation*, 6(2):182–197, 2002.

[3] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multi-objective optimization. *Evolutionary Computation*, 15(1):1–28, 2007.

[4] H. Li and Q. Zhang. Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transaction on Evolutionary Computation*, 13(2):284–302, 2009.

[5] A. Qin, V. Huang, and P. Suganthan. Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transaction on Evolutionary Computation*, 13(2):398–417, 2009.

[6] R. A. Sarker and T. Ray. *Agent-Based Evolutionary Search*. Springer, 2010.

[7] F. Stonedahly, W. Randyz, and U. Wilensky. Multi-agent learning with a distributed genetic algorithm: Exploring innovation diffusion on networks. In *Proceedings of the AAMAS Workshop on ALAMAS+ALAg*, 2008.

[8] Y. Wang, Z. Cai, and Q. Zhang. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Transaction on Evolutionary Computation*, 15(1):55–66, 2011.

[9] Y. Wang, J. Zhang, and J. Vassileva. Effective web service selection via communities formed by super-agents. In *Proceedings of the IEEE/WIC/ACM Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, 2010.

[10] J. Zhang and R. Cohen. A comprehensive approach for sharing semantic web trust ratings. *Computational Intelligence*, 23(3):302–319, 2007.

[11] W. Zhong, J. Liu, M. Xue, and L. Jiao. A multiagent genetic algorithm for global numerical optimization. *IEEE Transactions on System, Man, Cybernetics, Part B: Cybernetics*, 34(2):1128–1141, 2004.

[12] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. *Technical report 103, Computer Engineering and Networks Laboratory (TIK), ETH Zurich, Zurich, Switzerland*, 2002.

Table 3: Hypervolume Median and IQR of NSGAII, NSGAII-T, SPEA2, SPEA2-T, MOEA/D, MOEA/D-T

MOPs	NSGAII	NSGAII-T	SPEA2	SPEA2-T	MOEA/D	MOEA/D-T
ZDT1	6.60E-01±3.15E-04−	6.60E-01±3.66E-04−	6.62E-01±1.66E-04≈	6.62E-01±4.65E-05+	6.61E-01±2.08E-04−	6.62E-01±4.51E-07
ZDT2	3.27E-01±3.07E-04−	3.27E-01±2.91E-04−	3.28E-01±7.83E-05≈	3.29E-01±3.85E-05+	3.28E-01±1.15E-04−	3.28E-01±1.21E-08
ZDT3	5.15E-01±7.64E-05+	5.15E-01±1.12E-04+	5.16E-01±8.76E-05+	5.16E-01±2.71E-05+	5.14E-01±2.95E-05−	5.14E-01±8.61E-06
ZDT4	6.61E-01±1.83E-04−	6.60E-01±2.94E-04−	6.62E-01±5.06E-05+	6.62E-01±1.67E-01≈	6.61E-01±2.66E-04−	6.62E-01±1.05E-04
ZDT6	3.98E-01±3.84E-04−	4.00E-01±3.16E-04−	4.01E-01±2.48E-04−	4.01E-01±2.03E-05+	4.01E-01±2.50E-07−	4.01E-01±6.21E-09
DTLZ1	7.78E-01±4.39E-03+	7.76E-01±3.63E-03+	7.97E-01±5.31E-04+	7.97E-01±2.52E-04+	7.61E-01±5.58E-04−	7.61E-01±1.22E-04
DTLZ2	3.93E-01±4.23E-03≈	3.97E-01±4.28E-03+	4.19E-01±1.42E-03+	4.29E-01±8.00E-04+	3.92E-01±1.30E-03≈	3.93E-01±2.67E-04
DTLZ3	3.99E-01±7.80E-03+	3.96E-01±4.96E-03+	4.28E-01±1.42E-03+	4.29E-01±8.24E-04+	3.91E-01±2.12E-03−	3.93E-01±3.15E-04
DTLZ4	3.95E-01±3.33E-03+	3.94E-01±4.08E-03+	4.14E-01±1.33E-03+	4.22E-01±5.10E-04+	3.95E-01±2.38E-03+	3.88E-01±2.04E-04
DTLZ5	9.41E-02±1.20E-04+	9.41E-02±1.38E-04+	9.45E-02±1.13E-04+	9.47E-02±2.22E-05+	9.15E-02±1.24E-05−	9.15E-02±2.49E-07
DTLZ6	9.51E-02±1.49E-02+	9.46E-02±1.51E-04+	9.56E-02±3.57E-05+	9.56E-02±2.03E-05+	9.24E-02±2.63E-06+	9.23E-02±1.23E-07
DTLZ7	2.98E-01±2.71E-03+	3.04E-01±2.91E-03+	3.07E-01±1.67E-03+	3.12E-01±1.29E-03+	2.17E-01±2.27E-03≈	2.17E-01±2.75E-03
UF1	5.71E-01±1.59E-02−	6.29E-01±1.39E-02−	5.44E-01±2.63E-02−	5.73E-01±3.61E-02−	6.57E-01±2.26E-03≈	6.57E-01±1.34E-03
UF2	6.30E-01±7.40E-03−	6.48E-01±2.05E-03−	6.33E-01±8.49E-03−	6.42E-01±2.93E-03−	6.47E-01±9.69E-03−	6.56E-01±1.15E-03
UF3	4.67E-01±4.35E-02−	6.36E-01±1.18E-02−	4.37E-01±5.63E-02−	4.43E-01±2.93E-02−	6.37E-01±2.72E-02−	6.50E-01±8.98E-03
UF4	2.64E-01±1.17E-03−	2.79E-01±7.45E-03≈	2.71E-01±7.26E-04−	2.80E-01±5.16E-04+	2.27E-01±6.69E-03−	2.78E-01±9.98E-04
UF5	1.65E-01±1.31E-01≈	3.27E-01±4.94E-02+	1.87E-01±7.82E-02≈	1.29E-01±2.03E-01−	9.18E-02±1.52E-01−	2.00E-01±1.12E-01
UF6	2.32E-01±5.92E-02≈	2.33E-01±8.69E-02≈	2.44E-01±1.08E-01≈	2.25E-01±1.05E-01−	1.99E-01±1.35E-01−	2.51E-01±5.94E-02
UF7	4.43E-01±1.27E-01−	4.76E-01±3.68E-03−	4.36E-01±1.51E-01−	4.49E-01±1.26E-02−	4.88E-01±3.94E-03≈	4.88E-01±2.45E-03
UF8	2.05E-01±1.14E-01−	1.35E-01±1.57E-01−	1.56E-01±8.86E-03−	2.59E-01±9.33E-02−	2.84E-01±5.70E-03−	3.02E-01±3.12E-03
UF9	3.82E-01±1.11E-01−	3.58E-01±1.74E-01−	5.44E-01±9.88E-02−	5.98E-01±1.36E-02≈	5.32E-01±1.07E-01−	5.46E-01±1.07E-01
UF10	2.21E-02±4.52E-02−	3.76E-02±8.65E-02−	4.56E-02±3.72E-02−	1.14E-01±2.21E-02≈	4.08E-02±4.72E-02−	1.32E-01±7.59E-02
UF11	0.00E-00±0.00E-00−	0.00E-00±0.00E-00−	0.00E-00±0.00E-00−	0.00E-00±0.00E-00−	3.25E-05±9.28E-05−	3.08E-04±2.86E-04
UF12	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00
UF13	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00
CF1	4.66E-01±1.71E-03+	4.71E-01±8.97E-04+	4.62E-01±2.69E-03+	4.42E-01±5.23E-03−	4.39E-01±1.08E-05≈	4.59E-01±2.00E-02
CF2	5.88E-01±3.55E-02−	6.09E-01±1.87E-03−	5.72E-01±2.74E-02−	5.61E-01±3.05E-02−	6.50E-01±1.28E-03−	6.51E-01±1.19E-03
CF3	7.56E-02±6.04E-02−	1.46E-01±9.96E-02≈	1.03E-01±5.43E-02≈	0.00E-00±5.87E-02−	1.14E-01±3.21E-02≈	1.18E-01±5.46E-02
CF4	1.59E-01±9.84E-02−	2.02E-01±5.38E-02−	2.94E-02±1.70E-01−	7.55E-02±1.18E-01−	5.44E-01±7.82E-03−	5.49E-01±4.00E-03
CF5	0.00E-00±9.62E-02−	1.53E-01±2.18E-01−	0.00E-00±4.88E-02−	0.00E-00±0.00E-00−	3.17E-01±7.34E-02≈	3.38E-01±2.00E-01
CF6	3.38E-01±1.48E-01−	4.18E-01±1.28E-01−	2.36E-01±2.50E-01−	1.07E-01±2.25E-01−	6.44E-01±1.52E-02−	6.58E-01±1.96E-03
CF7	1.79E-01±2.38E-01−	2.66E-01±1.79E-01−	0.00E-00±1.65E-01−	0.00E-00±0.00E-00−	4.31E-01±7.78E-02−	4.69E-01±1.94E-01
CF8	0.00E-00±9.89E-02−	1.62E-01±5.95E-02−	2.17E-01±1.97E-01≈	2.00E-01±2.43E-01≈	2.24E-01±9.42E-02≈	2.04E-01±5.07E-02
CF9	2.03E-01±8.58E-02−	2.67E-01±2.05E-02−	2.85E-01±2.06E-02−	3.04E-01±1.96E-02−	3.31E-01±1.50E-02+	3.25E-01±2.91E-02
CF10	0.00E-00±0.00E-00−	0.00E-00±0.00E-00−	0.00E-00±0.00E-00−	0.00E-00±0.00E-00−	1.85E-01±3.78E-02−	2.07E-01±6.46E-02

+ , ≈ and − represent previous algorithm statistically significant better, similar and worse than the last algorithm, respectively

Table 4: Hypervolume Median and IQR of CMA-ES, SaDE, CoDE, MOEA/D-T3, MOEA/D-T', MOEA/D-T

MOPs	CMA-ES	SaDE	CoDE	MOEA/D-T3	MOEA/D-T'	MOEA/D-T
ZDT1	6.16E-01±1.16E-02−	6.62E-01±9.38E-06−	6.62E-01±1.22E-05−	6.62E-01±2.15E-06≈	6.62E-01±1.13E-08+	6.62E-01±4.51E-07
ZDT2	2.99E-01±2.26E-02−	3.28E-01±3.18E-06−	3.28E-01±3.28E-01−	3.28E-01±7.88E-09+	3.28E-01±3.05E-10+	3.28E-01±1.21E-08
ZDT3	3.90E-01±5.47E-02−	5.14E-01±1.23E-05−	5.14E-01±1.77E-05−	5.14E-01±2.73E-06≈	5.14E-01±4.86E-07+	5.14E-01±8.61E-06
ZDT4	0.00E-00±0.00E-00−	2.10E-01±2.08E-01−	0.00E-00±0.00E-00−	1.08E-01±2.10E-01−	2.75E-01±2.33E-01−	6.62E-01±1.05E-04
ZDT6	2.57E-01±3.83E-02−	4.01E-01±5.60E-07−	4.01E-01±8.23E-08−	4.01E-01±2.24E-08≈	4.01E-01±1.58E-09+	4.01E-01±6.21E-09
DTLZ1	0.00E-00±0.00E-00−	7.61E-01±1.63E-04+	1.06E-01±1.06E-01−	7.39E-01±6.55E-01−	7.61E-01±5.49E-05≈	7.61E-01±1.22E-04
DTLZ2	3.90E-01±4.01E-03−	3.93E-01±3.00E-04+	3.93E-01±2.77E-04+	3.93E-01±2.32E-04≈	3.93E-01±3.07E-04≈	3.93E-01±2.67E-04
DTLZ3	0.00E-00±0.00E-00−	0.00E-00±0.00E-00−	0.00E-00±0.00E-00−	0.00E-00±0.00E-00−	0.00E-00±0.00E-00−	3.93E-01±3.15E-04
DTLZ4	1.94E-01±3.06E-02−	3.89E-01±6.15E-04+	3.89E-01±5.20E-04+	3.88E-01±4.63E-04≈	3.88E-01±4.29E-04≈	3.88E-01±2.04E-04
DTLZ5	9.11E-02±5.47E-05−	9.15E-02±5.07E-07+	9.15E-02±6.82E-07≈	9.15E-02±1.50E-07≈	9.15E-02±2.50E-08≈	9.15E-02±2.49E-07
DTLZ6	0.00E-00±0.00E-00−	9.23E-02±6.59E-07≈	9.23E-02±8.99E-07≈	9.23E-02±5.58E-08≈	9.23E-02±1.18E-10≈	9.23E-02±1.23E-07
DTLZ7	2.28E-01±2.54E-02+	2.15E-01±2.34E-03≈	2.16E-01±2.55E-03≈	2.17E-01±2.65E-03≈	2.17E-01±2.61E-03≈	2.17E-01±2.75E-03
UF1	5.22E-01±2.23E-02−	6.40E-01±4.48E-03−	6.55E-01±2.12E-03−	6.55E-01±2.61E-03−	6.53E-01±6.08E-03−	6.57E-01±1.34E-03
UF2	6.24E-01±7.82E-03−	6.49E-01±5.45E-03−	6.53E-01±2.91E-03−	6.52E-01±4.31E-03−	6.49E-01±4.57E-03−	6.56E-01±1.15E-03
UF3	4.52E-01±2.42E-02−	5.16E-01±1.03E-01−	6.34E-01±2.19E-02−	6.46E-01±1.60E-02≈	6.19E-01±3.39E-02−	6.50E-01±8.98E-03
UF4	2.05E-01±3.66E-03−	2.81E-01±9.38E-04+	2.77E-01±6.80E-04−	2.79E-01±1.03E-03+	2.80E-01±1.99E-03+	2.78E-01±9.98E-04
UF5	0.00E-00±0.00E-00−	1.87E-01±1.07E-01≈	5.93E-02±1.27E-01−	1.72E-01±1.20E-01≈	2.34E-01±1.04E-01≈	2.00E-01±1.12E-01
UF6	7.49E-03±1.84E-02−	2.28E-01±5.13E-02−	2.14E-01±1.31E-01−	2.34E-01±4.18E-02≈	2.33E-01±6.99E-02−	2.51E-01±5.94E-02
UF7	1.84E-01±1.19E-01−	4.76E-01±5.49E-03−	4.86E-01±3.26E-03−	4.86E-01±2.64E-03−	4.84E-01±7.08E-03−	4.88E-01±2.45E-03
UF8	1.99E-01±1.63E-02−	2.91E-01±6.63E-03−	3.02E-01±3.91E-03≈	3.01E-01±5.89E-03≈	2.96E-01±8.34E-03−	3.02E-01±3.12E-03
UF9	4.69E-01±3.80E-02−	6.20E-01±1.13E-01≈	5.46E-01±1.10E-01≈	5.45E-01±1.07E-01≈	5.46E-01±1.06E-01≈	5.46E-01±1.07E-01
UF10	0.00E-00±0.00E-00−	1.86E-01±4.19E-02+	8.66E-02±7.76E-02−	1.14E-01±8.39E-02≈	6.25E-02±1.51E-01≈	1.32E-01±7.59E-02
UF11	3.18E-04±2.87E-04≈	1.80E-04±1.91E-04−	3.39E-04±2.15E-04≈	3.34E-04±2.88E-04≈	3.08E-04±2.86E-04≈	3.08E-04±2.86E-04
UF12	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00
UF13	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00≈	0.00E-00±0.00E-00
CF1	4.41E-01±2.00E-02−	4.39E-01±2.00E-02−	4.39E-01±2.00E-02≈	4.39E-01±2.00E-02≈	4.39E-01±2.00E-02−	4.59E-01±2.00E-02
CF2	5.72E-01±4.76E-02−	6.50E-01±3.79E-03−	6.50E-01±9.01E-04−	6.50E-01±1.52E-03−	6.50E-01±1.47E-03−	6.51E-01±1.19E-03
CF3	3.13E-02±4.62E-02−	1.44E-01±4.68E-02≈	1.32E-01±1.23E-01≈	1.45E-01±6.43E-02≈	1.33E-01±1.35E-01≈	1.18E-01±5.46E-02
CF4	3.67E-01±8.93E-02−	5.19E-01±1.07E-02−	5.44E-01±5.55E-03−	5.44E-01±4.30E-03−	5.40E-01±7.07E-03−	5.49E-01±4.00E-03
CF5	0.00E-00±0.00E-00−	2.91E-01±1.69E-01−	3.30E-01±1.68E-01≈	4.23E-01±2.65E-01≈	2.92E-01±2.04E-01≈	3.38E-01±2.00E-01
CF6	6.41E-01±1.18E-02−	6.52E-01±7.46E-03−	6.56E-01±3.13E-03−	6.56E-01±2.53E-03−	6.45E-01±3.12E-02−	6.58E-01±1.96E-03
CF7	0.00E-00±9.56E-03−	5.02E-01±4.71E-02≈	4.26E-01±2.16E-01−	5.39E-01±2.21E-01≈	4.07E-01±2.31E-01−	4.69E-01±1.94E-01
CF8	2.02E-01±4.72E-02≈	1.63E-01±8.09E-02−	1.71E-01±6.53E-02−	2.17E-01±5.64E-02≈	2.05E-01±6.89E-02≈	2.04E-01±5.07E-02
CF9	3.02E-01±2.67E-02−	2.78E-01±9.24E-03−	3.07E-01±4.08E-02−	3.15E-01±3.55E-02≈	2.89E-01±2.80E-02−	3.25E-01±2.91E-02
CF10	0.00E-00±0.00E-00−	2.02E-01±2.61E-03−	2.03E-01±2.37E-03−	2.07E-01±4.39E-04≈	2.07E-01±3.45E-04≈	2.07E-01±6.46E-02

+ , ≈ and − represent previous algorithm statistically significant better, similar and worse than the last algorithm, respectively

A qualitative reputation system for multiagent systems with protocol-based communication

Emilio Serrano
Facultad de Informática
Universidad de Murcia
Murcia, Spain
emilioserra@um.es

Michael Rovatsos
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, UK
mrovatso@inf.ed.ac.uk

Juan Botia
Facultad de Informática
Universidad de Murcia
Murcia, Spain
juanbot@um.es

ABSTRACT

We propose a novel method for assessing the reputation of agents in multiagent systems that is capable of exploiting the structure and semantics of rich agent interaction protocols and agent communication languages. Our method is based on using so-called *conversation models*, i.e. succinct, qualitative models of agents' behaviours derived from the application of data mining techniques on protocol execution data in a way that takes advantage of the semantics of inter-agent communication available in many multiagent systems. Contrary to existing systems, which only allow for querying agents regarding their assessment of others' reputation in an *outcome-based* way (often limited to distinguishing between "successful" and "unsuccessful" interactions), our method allows for contextualised queries regarding the structure of past interactions, the values of content variables, and the behaviour of agents across different protocols. Moreover, this is achieved while preserving maximum privacy for the reputation querying agent and the witnesses queried, and without requiring a common definition of reputation, trust or reliability among the agents exchanging reputation information. A case study shows that, even with relatively simple reputation measures, our qualitative method outperforms quantitative approaches, proving that we can meaningfully exploit the additional information afforded by rich interaction protocols and agent communication semantics.

Categories and Subject Descriptors

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

General Terms

Algorithms, Theory, Design

Keywords

Trust and reputation, agent communication, data mining

1. INTRODUCTION

Reputation, i.e. the beliefs or opinions generally held about other agents in a society, is one of the main means of evaluating the trustworthiness and reliability of individuals in multiagent systems (MASs). In the trust and reputation literature [5], trust is usually taken to denote the belief that a party will act cooperatively and not fraudulently, while reputation normally refers to trust information propagated through a social network of individuals [6]. The autonomy and heterogeneity of agents in open MASs makes the use of reputation in MASs particularly challenging, and often impedes the use of centralised trustworthy authorities such as the reputation models implemented in some internet-based markets, e.g. Amazon [1] or eBay [2] (although such reputation mechanisms are certainly most popular in the real world [5]). Yet, from the point of view of an agent, a correct assessment of others' reputation may greatly enhance performance, as it can be used to make appropriate decisions regarding which agents to interact with and how to behave in these interactions.

Existing trust and reputation approaches [5, 11, 8, 12] mostly focus on a purely *quantitative* assessment of trust, based on witness reports regarding positive/successful and negative/unsuccessful interaction experiences, usually only making binary (or one-dimensional numerical) distinctions resulting focussing on a single property of interactions that describes the trustworthiness or reliability of the target (i.e. reputation-evaluated) agent. Even when these methods allow for queries with a more "semantic" content (to ask for the reputation of an agent with regard to particular products, types of services, etc) [8], the assessment is always entirely *outcome*-oriented, and allows no assessment of the qualitative properties of the interaction process, i.e. the content and sequence of messages exchanged and physical actions observed.

Adopting this quantitative perspective effectively ignores the interaction mechanisms provided by many multiagent systems, in particular complex structure-rich interaction protocols that use agent communication languages (ACLs) with formal semantics. As opposed to low-level interaction mechanisms in other distributed systems, these languages and protocols attempt to capture shared meaning for messages exchanged in MASs, and the structure and "knowledge-level" assumptions captured in ACLs and interaction protocols is semantically rich and can be used to extract *qualitative properties* of observed conversations among agents.

In this paper, we introduce a novel reputation system based on the *qualitative context mining* approach proposed by Serrano *et al* [10], which allows us to exploit the semantics and structure of agent interactions, in order to produce better, contextualised assessments of reputation that can be tailored to the needs of the reputation-evaluating agent and

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

inform her interaction decisions. Our method is based on extracting succinct models of the evaluated agents' behaviour from previous interaction data. These can be queried by the evaluating agent (whether or not she is the *modelling* agent who has constructed the conversation model) with respect to specific protocols, paths within these protocols, or values of constraint arguments that are part of the protocol definition. Our approach minimises information disclosure among agents: The evaluating agent might request the entire conversation model from the modelling agent (which does not require the modelling agent to share her original interaction data, and thus also limits the bandwidth needed for data exchange) and perform queries herself on it (to avoid sharing definitions of what counts as "trustworthy" or "untrustworthy" to her). Alternatively, the evaluating agent can share these definitions of trustworthiness and query a modelling agent unwilling to transmit her conversation model, and only obtain reputation assessments in return, without access to the full conversation model.

What is more, through experiments in an example e-commerce scenario, we show that our reputation system is capable of effectively utilising the additional information provided by rich interaction protocols and ACLs, and results both in better predictions of future interaction behaviour of evaluated agents, and in improved responsiveness to unexpected changes in others' behaviours. This can be achieved by defining relatively straightforward reputation measures on top of the qualitative reputation assessment mechanism.

The remainder of the paper is structured as follows: Section 2 reviews the qualitative context mining approach suggested in [10] and describes how it is used as a basis for interaction data analysis in our system. In section 3, we introduce the proposed reputation measures that can be defined on top of our qualitative data analysis method. An empirical analysis of our method is presented in section 4. Section 5 discusses related work, and section 6 concludes.

2. MINING AGENT CONVERSATIONS

As described above, our reputation system uses the framework proposed in [10] as a base method for interaction analysis. The context mining approach presented there does not assume a specific protocol or agent communication language for MASs, but represents protocols in a very general way as graphs whose nodes are speech-act like messages placeholders, and whose edges define transitions among messages that give rise to message sequences specified as admissible according to the protocol. The edges are labelled with logical constraints, i.e. formulate logical conditions that the agent using the protocol is able to verify. These act as guards on a given transition, so that the message corresponding to a child node can only be sent if the constraint(s) along its incoming edge from the parent node (the message just observed) can be satisfied.

[10] defines a *protocol model* as a graph $G = (V, E)$ where nodes $v \in V$ are labelled with messages $m(v) = q(X, Y, Z)$, q is a performative and X, Y, and Z are sender/receiver/-content variables, respectively. Edges are labelled with a (conjunctive) list of logical constraints

$$c(e) = \{c_1(t_1, \ldots, t_{k_1}), \ldots, c_n(t_1, \ldots t_{k_n})\}$$

where each constraint $c_i(\ldots)$ has arity k_i, head c_i and arguments t_j. Constraints can be arbitrary logical formulas composed of predicates which may contain constants, functions or variables, with all variables implicitly universally quantified. It is assumed that all outgoing edges of a node result in messages with distinct performatives, i.e. for all $(v, v') \in E$, $(v, v'') \in E$

$$(m(v') = q(\ldots) \wedge m(v'') = q(\ldots)) \Rightarrow v' = v''$$

so that each observed message sequence corresponds to (at most) one path in G by virtue of its performatives. Figure 1 shows an example protocol model in this generic format.

The *semantics* of a protocol model G is based on considering finite paths $\pi = v_1 \stackrel{e_1}{\rightarrow} v_2 \stackrel{e_2}{\rightarrow} \ldots \stackrel{e_{n-1}}{\rightarrow} v_n$ in the graph G (which may include unfoldings of cycles, assuming fresh variable names each time a node is revisited). If $\mathbf{m} = \langle m_1, \ldots, m_n \rangle$ are the ground messages observed in a run, $G(\mathbf{m}) = \langle \pi, \theta \rangle$ returns the (unique) path π that can be traced in G following the observed messages, and θ is the most general unifier of the set

$$\{m_1, \ldots, m_n\} \cup \{m(v_i) | 1 \leq i \leq n\}$$

and $\pi = v_1 \stackrel{e_1}{\rightarrow} \ldots \stackrel{e_{n-1}}{\rightarrow} v_n$. In other words, the pair $\langle \pi, \theta \rangle$ returns the path and variable substitution the message sequence $\mathbf{m}$ corresponds to in protocol model G. While context models are defined in [10] based on an analysis of the logical formula resulting from constraints along a path, for our purposes it is sufficient to consider pairs $\langle \pi, \theta \rangle$ that correspond to message sequences $\mathbf{m}$ of past observed interactions as samples for data mining algorithms.

To explain how we proceed in collecting and processing samples of protocol executions, consider the protocol model shown in figure 1. An execution run using this model will consist of a sequence of messages and constraints satisfied along that path (or, at least, presumably satisfied, assuming that the other agent only utters a message when its preconditions are satisfied) and will be translated to a list of feature-value pairs where the features are variables used in the messages, and the values their respective ground instantiations. In terms of actual data mining methods used, we restrict ourselves here to decision tree learning (we use *J48*, an open source implementation of the C4.5 algorithm [3]). Though [10] compares several other techniques, our system operates on trees like the one shown in the example of figure 2 obtained from the protocol in figure 1. As in our evaluation in section 4, this is derived from a scenario where agents use the protocol to negotiate over cars using a well-known database for car evaluation [4]. In this scenario, the modelling agent (who builds the tree from past data) is a potential customer (role A) who has requested offers from a car selling agent (role B) where T specifies the technical characteristics of the car, including number of doors, capacity in terms of persons to carry, the size of the luggage boot, the estimated safety of the car, price and maintenance cost. We assume that a feature vector for terms is of the form

$$T = (doors, persons, lug\_boot, safety, price, maint)$$

where

$doors \in \{2, 3, 4, 5\text{-}more\}$	$persons \in \{2, 4, more\}$
$maint \in \{v\text{-}high, high, med, low\}$	$safety \in \{low, med, high\}$
$price \in \{v\text{-}high, high, med, low\}$	$lug\_boot \in \{small, med, big\}$

The conversation model shown in figure 2, for example, shows that seller S_8, for instance, performed 44 successful negotiations but also that these involved cars with a low maintenance cost, medium safety, and a low buying price.

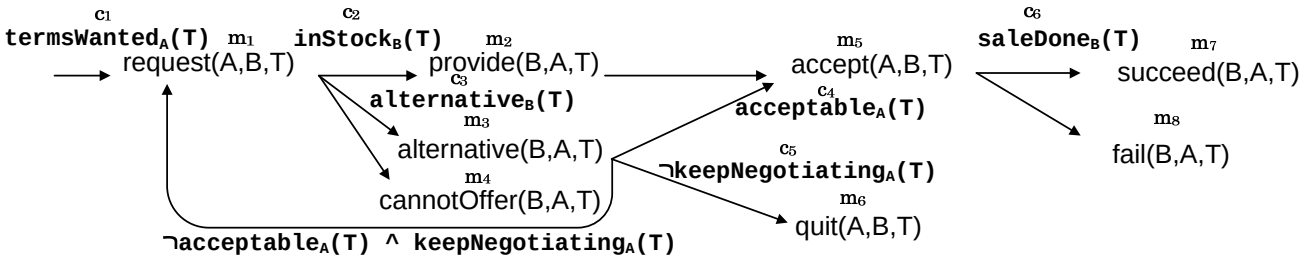


Figure 1: A simple negotiation protocol model: A requests a product with description T (the *terms*) from B. The initial response from B depends on availability: if terms T cannot be satisfied, A and B go through an iterative process of negotiating new terms for the item, depending on the *keepNegotiating*, *acceptable*, and *alternative* predicates (for simplicity, we use a fixed variable T in the diagram, although in the course of a negotiation its value may change). In case of acceptance (which implies payment), B may succeed or fail in delivering the product. Edge constraints are annotated with the variable representing the agent that has to validate them. Additional (redundant) shorthand notation c_i/m_j is introduced. Different out-edges represent XOR if constraints are mutually exclusive.

```
maint = v-high: F (47.0)
maint = high: F (48.0)
maint = med: F (299.0)
maint = low
|   safety = low: F (88.0/2.0)
|   safety = med
|   |   B = S_1: F (17.0/1.0)
|   |   B = S_2: F (15.0)
|   |   B = S_3
|   |   |   price = v-high: S (0.0)
|   |   |   price = high: S (0.0)
|   |   |   price = med: F (3.0)
|   |   |   price = low: S (56.0)
|   |   B = S_4: F (21.0)
|   |   B = S_5: F (13.0)
|   |   B = S_6: F (13.0)
|   |   B = S_7: S (62.0/1.0)
|   |   B = S_8
|   |   |   price = v-high: S (0.0)
|   |   |   price = high: S (0.0)
|   |   |   price = med: F (4.0)
|   |   |   price = low: S (44.0)
```

Figure 2: J48 output for 1000 negotiations. The notation `a =v : S/F` denotes that "if `a` has value `v` the target predicate has value `S/F`". Every leaf includes the number of instances classified in parentheses (the second number appearing to the right of the "/" in some cases is the incorrectly classified instances.

In what follows we shall assume, somewhat informally, that a conversation model has the form of such a tree which can provide path information for (potentially incomplete) sets of variable-value pairs, and denote such tree structures generically as *conversation models CM*. In principle, many other formalisms can be conceived of that achieve the same, such as a relational database, a set of Horn clauses, a Bayesian classifier, etc.

3. REPUTATION SYSTEM

As suggested in the introduction, our reputation system includes an *evaluating agent* a who is trying to assess the reputation of the *target agent* b using a conversation model provided by a *modelling agent* (or *witness*) m, who may, but need not be, the same agent as a.

3.1 Querying the modelling agent

Three modes of reputation calculation are possible in principle: (i) a obtains the entire conversation model from m which has been built by m based on a's definitions of success and failure, and then makes specific queries for specific instances (i.e. lists of variable substitutions) in the model, (ii) a is not granted access to the conversation model, but instead sends only information about its definition of success and failure to m and then m answers particular queries of a regarding specific instances, or (iii) a receives the interaction data from m and builds the conversation model herself.

In our system, we use a method that allows for a uniform treatment of all three cases. This is achieved by splitting the querying process into two steps: *providing path classification*, where a informs m of which paths in the protocol model it considers successful and which are deemed unsuccessful, and m builds its classifier using the methods described in the previous section to build the conversation model; and *instance querying*, where a sends m a specific (though potentially partial) substitution for variables occurring in the model, and m returns a success/failure prediction based on the conversation model previously constructed. With this, whether case (i) or (ii) applies makes no difference from an algorithmic point of view – the same two processing steps are performed regardless of who holds the model. Moreover, since the path classification of a is probably stable over time, whereas instance queries vary (and occur more often), it makes sense to avoid rebuilding the conversation model unless path classification changes, and issue instance queries to the model that only rarely changes (except when m wants to rebuild it based on new data, or is asked for updating it by a). Case (iii) can be basically ignored, as it simply amounts to $a = m$ (in all other cases nothing can be really gained from sending around the entire dataset, methods (i) or (ii) are preferable, at least as long as m is trusted).

Path classification requires that a send m a set of successful paths $\mathcal{S} \subseteq (E \times V)^+$ in protocol model $G = (V, E)$,

and we write $CM(G, \mathcal{S})$ (or simply CM, where G and $\mathcal{S}$ are assumed to be specified) for the conversation model derived by m adding an additional *Outcome* to each path $s \in \mathcal{S}$ with value S (for success) and F (for failure) to all paths $s \notin \mathcal{S}$. The reason we allow for a set of paths to be specified as successful, is that various types of untrustworthy behaviour might occur. In our example protocol, B might claim to provide terms that are not in stock, she might propose alternatives unrelated to the terms proposed by A, might provide terms in the final message unrelated to those accepted by A, or simply offer unacceptable terms such as an excessive price. Even in simpler cases, e.g. when identifying those paths as successful which terminate with a `succeed` message, one may need to specify relatively complex rules that involve entire sets of paths like the following:

if $(\overset{c_1}{\rightarrow} m_1((\overset{\neg c_2 \wedge c_3}{\rightarrow} m_3 \overset{\neg c_4 \wedge c_5}{\rightarrow} m_1)^* \overset{\neg c_2 \wedge c_3}{\rightarrow} m_3 \overset{c_4}{\rightarrow})|$
$(\overset{c_2}{\rightarrow} m_2 \rightarrow))m_5 \overset{c_6}{\rightarrow} m_7)$ **then** $Outcome = S$
else $Outcome = F$

Instances i queried for are lists of attribute-value pairs $i = \{V_1 = g_1, \ldots, V_n = g_n, Outcome = g\}$ for variables V_i occurring in the messages and constraints of protocol model G with ground values g_i from their respective domains in previous interactions, extended by the outcome value $g \in \{S, F\}$ for the queried instance. Querying for i basically amounts to asking "if $V_1, \ldots, V_n$ have values $g_1, \ldots, g_n$, will the outcome of the interaction be g?"

In our example conversation model, an instance query about target agent b concerning a successful outcome in a negotiation after asking for a a car with high safety assessment and low price is:

$$i = \{B = b, safety(T) = high, price(T) = low, Outcome = S\}$$

where we use functions like $safety(T)$, and $price(T)$ to return the respective values of the "terms" variable T. It should be noted that such queries neither need to contain all variables on the paths involved, nor that those paths provided in S need to terminate in leaves. Using CM instead of a simple database of past interaction data provides this flexibility.

3.2 Reputation and reliability

The basic reputation measure used by evaluating agents a in our system is defined as follows:

$$R(CM, i) = \begin{cases} 1 & \text{if } prediction(CM, i) = i.Outcome \\ -1 & \text{else} \end{cases}$$

where $prediction(CM, i)$ returns the classification value (S/F) from the conversation model CM given i. For this, the conversation model CM is used to classify the expected result of the interaction in i and if the predicted class matches the outcomes queried for by i, the prediction 1 (=correct) is returned. Note that, while we have assumed a binary good/bad classification in our formalisation, using a larger number of distinctive labels is straightforward, and even a numerical assessment would be possible using alternative data mining methods (such as a Bayes' Net). Note also that this simple measure already allows a to specify what it views precisely as "trustworthy", and that the same interaction data store can be queried by different evaluating agents easily without a shared notion of reputation. Moreover, G may contain a number of different (independent) protocols, and if different variables or constraints occur across several of these, all past interaction experience will be taken into account when building CM and can be queried simultaneously.

It is straightforward to generalise this measure to return values for a set of target agents $\mathcal{T}$ simply by extending the above function canonically to return a vector of values, taking into account appropriate substitutions:

$$R(CM, V, i, \mathcal{T}) = \langle R(CM, i_{V/b_1}), \ldots, R(CM, i_{V/b_n}) \rangle$$

where $\mathcal{T} = \{b_1, b_2, \ldots, b_n\}$ are the possible target agents and i_{V/b_j} is the extension of the instance query i by the assignment $V = b_j$ and V is the variable in G that refers to the role for which we want to evaluate the reputation of agent b_j. In our example above, if $i = \{maint(T) = low, safety(T) = med, price(T) = low, Outcome = S\}$ and $\mathcal{T} = \{s_1, s_2, s_3\}$, we would obtain $R(CM, V, i, \mathcal{T}) = \langle -1, -1, 1 \rangle$ as a prediction vector for the three agents in the seller's (B's) role. Such a query can be easily used to pick appropriate interaction partners from a set of agents.

To assess the reliability of a prediction provided by the conversation model, we also need to take into account how many past experiences match the query and what proportion of them has been correctly or incorrectly classified according to a rule in the conversation model. Here, it is important to restrict the set of correct/incorrect classifications to those queried by the evaluating agent. For example, assume the queried instance is

$$i = \{B = b, safety(T) = high, price(T) = low, Outcome = S\},$$

and the result of the prediction is S. The result of the reputation query would be $R(CM, i) = 1$, and a possible rule in the tree used for this prediction may have been "if $B = b$ and $safety(T) = high$) then $Outcome = S$". However, the instances that match the antecedent are a superset of those considered by the query, so that the number of correctly classified instances for this rule is an upper bound for those matching the query. To account for this, let $CM(i)$ the set of all rules in CM that match *at least* query i (i.e. they may contain more, but no less attribute-value pairs), and define the reliability of a reputation assessment as

$$r(CM, i) = \begin{cases} \frac{\sum_{\rho \in CM(i)} cci(\rho)}{\sum_{\rho \in CM(i)} ci(\rho)} & \text{if } \sum_{\rho \in CM(i)} ci(\rho) \neq 0 \\ 0 & \text{else} \end{cases}$$

where $ci(\rho)$ are the instances classified by rule ρ, and $cci(\rho)$ returns the number of correctly classified instances by the same rule. In figure 2 these numbers are shown adjacent to the leaves of the tree. This effectively evaluates the confidence of CM in its prediction by calculating the ratio of correctly classified samples that match the query compared to all matching samples in the modelling agent's data set.

3.3 Individual and collective reputation

Next, we can easily combine reputation and reliability to obtain the reputation by *personal experience* and by *group experience* measures used in reputation systems like [8]

$$PE(CM, i) = R(CM, i) \cdot r(CM, i)$$

as the product of reputation and reliability obtained for a simple query. If the instance i does not include an instantiation of the target agent, we can extend this, as before, to sets $\mathcal{T} = \{b_1, \ldots, b_n\}$ of target agents:

$$PE(CM, i, V, \mathcal{T}) = \langle R(CM, i_{V/b_1}) \cdot r(CM, i_{V/b_1}), \ldots, R(CM, i_{V/b_n}) \cdot r(CM, i_{V/b_n}) \rangle$$

Considering $|\mathcal{M}|$ modelling agents $m_1, m_2, \ldots, m_{|\mathcal{M}|}$, each of whom has a respective CM_j at her disposal built using the classification requirements provided by the evaluating agent, reputation by group experience is defined as[1]:

$$GE(\mathcal{M}, i) = \frac{\sum_{1 \leq j \leq |\mathcal{M}|} PE(CM_j, i)}{\sum_{1 \leq j \leq |\mathcal{M}|} r(CM_j, i)}$$

Here the modelling agents are used as witnesses who each provide a personal experience for the target query, and the evaluating agent normalises their individual reports by their respective reliabilities. Again, this can be extended to return a vector of values if the target agent is not specified in i:

$$GE(\mathcal{M}, i, V, \mathcal{T})[k] = \frac{\sum_{1 \leq j \leq |\mathcal{M}|} PE(CM_j, i)[k]}{\sum_{1 \leq j \leq |\mathcal{M}|} r(CM_j, i)}$$

where $1 \leq k \leq |\mathcal{T}|$.

With these, we can now define our main measure of *social reputation* as follows:

$$SR(\mathcal{M}, i) = \xi \cdot PE(CM_a, i) + (1 - \xi) \cdot GE(\mathcal{M}, i)$$

where ξ can be used to weight the impact of personal vs. group experience in the overall judgement. As above, in its vector form covering a set of target agents $\mathcal{T}$, social reputation is defined as

$$\begin{aligned} SR(\mathcal{M}, i, V, \mathcal{T})[j] = & \xi \cdot PE(CM_a, i, V, \mathcal{T})[j] + \\ & (1 - \xi) \cdot GE(\mathcal{M}, i, V, \mathcal{T})[j] \end{aligned}$$

for $1 \leq j \leq |\mathcal{T}|$. Note that ξ is effectively the only parameter introduced in our system that may be specific to a particular implementation. All other elements of the measures introduced above are generic. It should be remarked that as some popular rival approaches [5, 8], we do not include measures in the calculation of SR that take into account how much the witnesses are trusted (in terms of past interactions with them, not assessments of third parties), or the opinion toward a "group" the target agent belongs to. These could be easily defined in our framework, as discussed in section 5. As we show below, we can achieve good predictability without them, by focussing more on the structure and semantics of interactions in analysing past interactions.

4. EVALUATION

To illustrate the usefulness of our approach, we conducted a number of experiments in the simulated car selling domain introduced in section 2. Our scenario contains six preference profiles P_i for customer agents regarding T. These are used to define what cars are considered acceptable by the customers, and are specified as disjunctions of combinations of product properties, e.g.

$$\begin{aligned} P_1(T) = & (\mathit{persons} = \mathit{more} \wedge \mathit{lug\_boot} = \mathit{big} \wedge \mathit{price} = \mathit{low} \wedge \mathit{maint} = \mathit{low}) \vee (\mathit{persons} = \mathit{more} \wedge \mathit{lug\_boot} = \mathit{big} \wedge \mathit{price} = \mathit{med} \wedge \mathit{maint} = \mathit{med}) \vee (\mathit{doors} = \text{5-}\mathit{more} \wedge \mathit{persons} = \mathit{more} \wedge \mathit{price} = \mathit{low} \wedge \mathit{maint} = \mathit{low}) \vee \\ & (\mathit{doors} = \text{5-}\mathit{more} \wedge \mathit{persons} = \mathit{more} \wedge \mathit{price} = \mathit{med} \wedge \mathit{maint} = \mathit{med}) \end{aligned}$$

[1]As in the definitions of other measures below, we set this quantity to 0 when the denominator is 0 and omit this case for brevity.

We implement fifty customer agents C_1 to C_{50} with associated profiles $C_i \leftarrow P_{i \bmod 6}$, so that agents C_1 and C_7 use P_1, C_2 and C_8 use P_2, and so on.

Similarly, we specify three seller agent preference profiles Q_j, again specified in terms of T. These describe what types of cars a seller can offer. Additionally, every disjunction is labelled with *tb* or *ub* to indicate in which cases the seller will behave in a trustworthy or untrustworthy way when it negotiates those products. Again, we only show one of these profiles for illustration:

$$\begin{aligned} Q_1(T) = & (\mathit{safety} = \mathit{med} \wedge \mathit{price} = \mathit{low} \wedge \mathit{maint} = \mathit{low}) \rightarrow \mathit{tb} \\ & \vee (\mathit{safety} = \mathit{high} \wedge \mathit{price} = \mathit{low} \wedge \mathit{maint} = \mathit{low}) \rightarrow \mathit{tb} \\ & \vee (\mathit{safety} = \mathit{high} \wedge \mathit{price} = \mathit{med} \wedge \mathit{maint} = \mathit{med}) \rightarrow \mathit{ub} \end{aligned}$$

This profile specifies that the seller will respond positively to a request for terms ($\mathit{safety} = \mathit{med} \wedge \mathit{price} = \mathit{low} \wedge \mathit{maint} = \mathit{low}$), and that she will then also comply with all subsequent steps until the sale is completed. In those cases labelled *ub*, the seller will initially agree to the terms, but will then choose a random "failure" path in her subsequent behaviour. Our system implements 10 sellers S_1 to S_{10}, with associated profiles $S_j \leftarrow Q_{j \bmod 3}$.

4.1 Model construction and reputation measurement

To convert raw sequences of message exchanges to training data samples, we make the following design choices: As far as variables occurring in constraints are concerned, we uniformly record all attributes contained in "terms" descriptions T, including a "?" (unknown) value for those not mentioned in a given execution trace. This is feasible in the given protocol model as the amount of unspecified data is manageable. Our strategy to deal with loops is to only record the last value of every variable occurring in multiple iterations over the `alternative-request` sub-sequence for negotiation, as we are primarily interested in the final offer accepted or rejected by the customer.

The strategy that customer agents follow using our reputation system is explained below:

1. Each customer from $\mathcal{M} \in \{C_1, \ldots, C_{50}\}$ computes
$$SR(\mathcal{M}, i, B, \{S_1, \ldots, S_{10}\})$$
with $\xi = 1/50$ (i.e. equal weight is given to personal experience as to each of the 49 witnesses) and for a query i complying with one of their acceptable preferences in P_i. Each C_i thereby uses her own conversation model, built using only own the agent's own interaction experience.

2. Each customer chooses the seller S_j with the highest positive reputation value and interacts with that agent in the current negotiation. If the prediction of the model does not match the observed interaction experience, the agent re-builds her model from scratch.

3. If there is no such agent, the terms i are updated according to the customer's preferences, and we repeat from 1 (the disjunctive clauses in the P_i profiles are incomplete and can be easily randomly extended to obtain a specific requested car). If no seller with positive reputation can be identified after up to 100 attempts, the customer will interact with a random seller.

It should be observed that we deliberately test our system in a very "heavy" form of its usage, repeating the data mining step over all past interaction data, posing up to 100 queries until a positive prediction is returned, and using each customer agent as an independent modelling agent and witness for every other agent. This approach is chosen to illustrate that even this resource-intensive way of employing our method results in reasonable computation times, as will be shown below. This configuration also allows us to show the workings of our method in the "optimal" case, i.e. when investing a maximum effort of computation.

We compare the prediction accuracy of our system against a number of alternative reputation strategies, measured as the percentage of successful interactions over time:

Random. The seller is chosen randomly – this provides a baseline for the minimum performance that could be achieved without any use of reputation. An optimal strategy is not included, as 100% success constitutes the upper bound of what can be achieved in this scenario (we ensure that there are always sellers in the system who can provide the requested items in a trustworthy way).

Quantitative. The seller is chosen using a distance function based on the number of past successes and failures with them in the customer's personal experience. The function used is $D(s, f) = 1 - (1 + \frac{s}{2f+1})^{-1}$, where s is the number of successes and f the number of failures with a particular seller, and the seller to interact with is chosen with probability corresponding to $D(s, f)$ [9].

Personal experience only. Our reputation system is used as described above but with $\xi = 1$, i.e. the customer only takes her own interaction experiences into account. This method is chosen for comparison to assess the relative importance of witness information as compared to local interaction experience.

Restricted qualitative. Instead of structural and semantic information we use only A, B, and the *Outcome* label (S or F) in combination with the data mining technique. This serves to illustrate the performance of using the same data mining technique without any in-depth information about the content of interactions.

4.2 Static seller behaviour

As figure 3 shows, the results show that after 100 negotiations (2 negotiations per customer) all strategies exhibit similar performance. After 1000 negotiations (20 negotiations per customer) our reputation system greatly outperforms all other strategies, with the social reputation strategy converging much faster to optimal performance than the strategy based on personal experience only. This difference is understandable, as the conversation models combined in the social reputation strategy are based on a much broader variety of data earlier on in the process. However, later convergence of the "personal experience only" strategy also shows that it performs equally well in the long term, provided sufficient data becomes available. The plot also shows that a data mining approach without an analysis of the detailed structure of interactions does not perform any better than the purely quantitative approach, thus proving that the advantage of our method is indeed brought about by the inclusion of qualitative interaction properties rather than by the effectiveness of the data mining algorithm itself.

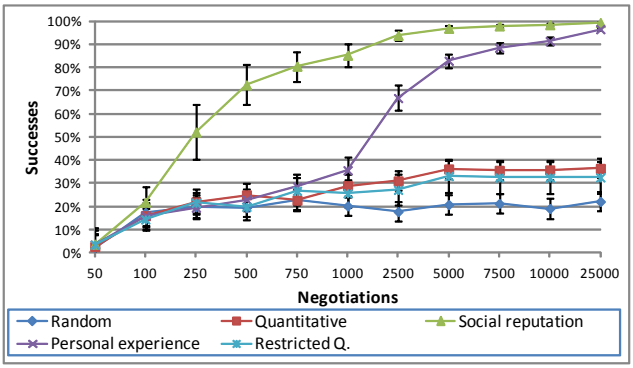


Figure 3: Average number of successful negotiations over number of total negotiations across all customers (100 experiments); error bars show standard deviation

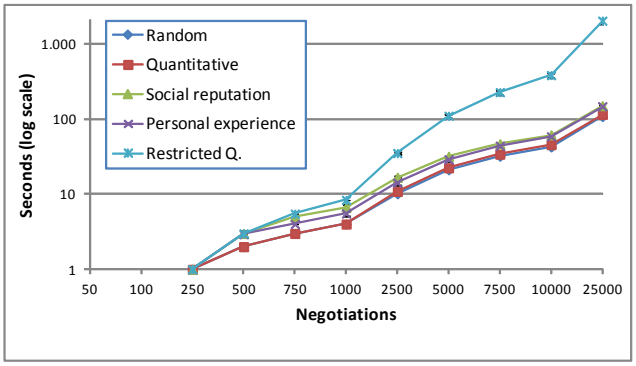


Figure 4: Time per negotiation, log scale (averaged over 100 experiments); the standard deviation across experiments is negligible

The downside of our method is of course increased runtime, at least when, as above, conversation models are rebuilt every time a customer obtains a wrong prediction, which happens very often, while also the datasets over which the models are built increase over time. Figure 4 shows the time taken on average per negotiation, which reaches around 150 seconds after 25000 negotiations for the personal and group experience methods. While this is clearly a shortcoming of our method, it is highly customisable in that the maximum amount of data processed or the frequency with which models are re-built can be adapted as suits the system designer (albeit at the cost of lower accuracy). Also, the runtime per negotiation is still much lower than the over 2000 seconds required by the restricted qualitative approach, which has to rebuild the model very often due to its failures. This also shows that a data mining-based analysis which doesn't take the semantic and structural dimension of communication into account actually combines the worst of both the quantitative (low performance, hence constant need to re-build model) and qualitative (high computational effort to rebuild model) worlds. It would only work well if a given seller behaved well or badly in *every* interaction.

4.3 Dynamic seller behaviour

The ability to respond to dynamic changes in others' behaviours is an important performance characteristic of reputation systems. In our second experiment, we introduce seller agents who suddenly switch their behaviour (from trustworthy to untrustworthy and vice versa) as specified in their original profiles (each rule resulting in *tb* will be modified to *ub* and vice versa). We compare the success rate of the following strategies for responding to dynamic behaviour change against the extreme cases ("no change" in seller behaviour to respond to, and "no strategy" to respond to changes in seller behaviour, i.e. fixed social reputation):

1. *Incongruence detection.* This method is based on erasing all previously collected data samples if a new prediction result is incorrect *and* there is past experience for same instance which provided the correct prediction. The idea behind this is that this should not happen unless evidence shows that the behaviour of the target agent(s) has changed drastically. The method requires that past queries are remembered, and may also lead to removal of many past data samples.

2. *Timestamp weighting.* The second strategy is based on weighting past samples according to their recency during model construction. A weight function $W : \mathbb{N}^2 \to [0, 1] \subset \mathbb{R}$ is employed which uses the current time stamp t and the time t' an instance was observed as a weight $W(t, t')$ for an interaction observed in the past. We use the same weight function as [8], i.e. $W(t, t') = t'/t$ to give more weight to samples closer to t.

3. *Weighted resampling.* Similar to the previous method, this strategy applies a re-sampling step after fixing the weights, i.e. it produces a random subsample of the dataset using sampling with replacement to produce a constant-sized dataset, where the selection probability is proportional to the sample weight [3].

4. *Fixed window.* This strategy simply retains a window with the last 1000 samples for model construction, omitting all previous samples. Another strategy has been added for a window with 500 samples instead of 1000.

The results for the different strategies are shown in figure 5. The plot shows that the incongruence detection strategy achieves the fastest recovery from the intermittent drop in success rate after the seller's behaviour change and manages to return to near-optimal performance very soon. As incongruence recovery strongly relies on an understanding of the structure of qualitative queries, this result illustrates that our reputation system not only manages to exhibit responsiveness (which can aid agents much in adapting to shifting behaviour of malicious agents who try to "massage" them into thinking they are trustworthy) with relatively simple dynamic re-evaluation strategies, but also that the qualitative approach we take is essential to enable such strategies.

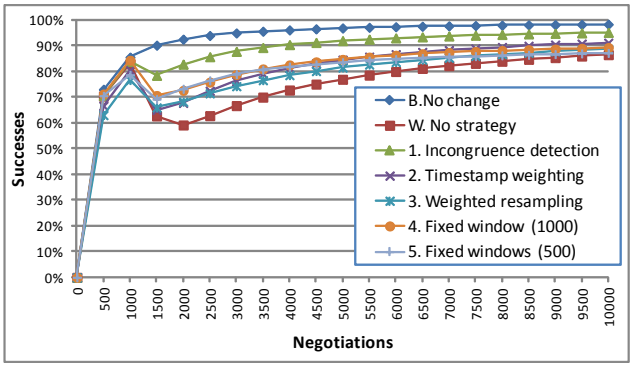


Figure 5: Success probability against number of negotiations, with dynamic seller behaviour change in round 1000 (average over 100 experiments).

5. RELATED WORK

Apart from systems that rely on a purely centralised reputation mechanism such as [1, 2, 13], popular and comparable recent distributed approaches include TRAVOS [11], Referral System [12] and FIRE [5]. All of them use two main information sources to compute reputation values, the *personal experience*, and *witness' experience*. FIRE [5] also uses additional dimensions, i.e. role-based rules and third-party references provided by the target agents. We argue that the major limitation of these approaches is that their definitions of reputation do not depend on the semantics of the domain and the structure of interactions. As a consequence, reputation values are only relevant when all evaluating agents are interested in the same aspect or type of interaction, and the modelling agent(s) calculate reputation based on precisely this information. To use the example of [6], if agents buy something on eBay, trusting a seller agent implies that she will send the right product to the right place at the right time. While some buyers will accept a small delay, they are not able to query the reputation system for such specific details. However, when all providers offer the same product with the same characteristics, as in the trading system used to assess FIRE, these approaches work well.

A notable exception to the lack of semantics in reputation approaches is the REGRET model proposed by Sabater and Sierra [8]. REGRET uses ontologies to detail the type of trust required by the evaluating agent, which can be used to query witnesses. REGRET's contribution is that, since the meaning of trust can be different for each agent, the evaluating agent must be able to ask to what extent witness agents trust the target agent concerning specific aspects of the interactions. Following our eBay example, this means that an agent can query the seller's reputation with regard to getting a low price, quick delivery, etc. Based on these values, the evaluating agent can define its own concept of global trust, e.g. giving less weight to price than to delivery. The main improvement of our approach over REGRET is that reputation is defined as a *model of behaviour* with arbitrarily complex properties, modelled on the basis of the interaction procedures used by the agents in a system. This allows agents to make much more informed decisions based on more fine-grained and flexible queries, makes a priori agreement on a set of specific ontological dimensions of trust across the system unnecessary, and also implies more concise reputation models that are not merely constantly growing databases of past interactions, but store regularities in observed behaviour in succinct data structures.

A limitation that we share with other approaches is that witnesses are assumed to be trustworthy. Although dealing

with untrustworthy witnesses is beyond the scope of this paper, our method provides improved capabilities which could be used to address this issue: When complete reputation models are exchanged between modelling and evaluating agent, the evaluating agent can assess the *long-term* reliability of a model by evaluating its reliability over its own past interaction experiences *prior* to using a prediction provided by this model to make concrete interaction decisions. Contrary to non-qualitative methods, this can be done *without* requiring access to the original interaction data the model was built with. Another possible strategy which illustrates the generality of our approach would be to model interactions with witnesses *themselves* as protocols, and build a trust model for them in much the same way as this is done for target agents.

The obvious weakness of our contribution are its complexity and requirement for additional knowledge. The definition of protocols, application of data mining algorithms, manipulation of conversation models, etc are much more elaborate and less efficient than the application of polynomial-time mathematical operations used in quantitative reputation systems. Possible measures to reduce the number of conversation models created are: (i) the use of data mining techniques which incorporate new experiences without rebuilding the entire model (incremental learning algorithms) [7], and (ii) not creating a new conversation model if this model is not expected to be better than the previous one. With this respect, one way of limiting the amount of computation performed is to rebuilds a conversation model only if a new experience is incorrectly classified by the old conversation model, or if the evaluating agent changes the set of classification rules which determine the classes of the instances before obtaining the conversation model.

6. CONCLUSION

In this paper, we have proposed a novel *qualitative* approach to reputation systems based on mining "deep models" of protocol-based agent interactions. Contrary to most existing methods, the reputation measures we define do not solely rely on the assessment of the predicted *outcome* of an interaction, but take the complex, knowledge- and content-rich structure and semantics of multiagent protocols and agent communication languages into account. On the side of the reputation-evaluating agent, this allows us to introduce more complex, fine-grained, and contextualised queries that can be posed to a reputation-modelling (collection of) witness(es), which results in higher prediction accuracy than quantitative methods as the queries are tailored to the needs of the agent. As a side-effect, our system also allows more intelligent and rationally reasoning agents to exploit the expressiveness our framework affords: As our case study shows, if agents have preferences and objectives specified in a language that can be related to the semantics of a protocol language, the reputation queries can be seamlessly constructed on the basis of their internal beliefs and mental states. On the side of the witness, our method leads to more concise, generalised models of target agents' behaviours, reducing the need to store huge amounts of past interaction data in what would otherwise be a "flat" database of past interactions, allows for disclosure of the model instead of transmission of primary interaction experience (which may also be subject to confidentiality restrictions), and enables different levels of privacy toward a reputation-querying agent without the need to modify the algorithms used to measure reputation. Our empirical results show that our method is capable of exploiting the additional structure and semantics we provide it with, both in terms of achieving higher prediction accuracy (sooner), and in terms of responding to unexpected changes in target agents' behaviours.

In the future, we would like to explore more elaborate data mining techniques, in particular to learn logical theories of the constraint definitions other agents apply from past interaction data, to evaluate our system in larger scenarios with a broader variety of interaction protocols and behaviour types, and to explore issues of trust in witnesses in order to be able to accommodate scenarios where witnesses are not necessarily trustworthy, or might even collude with target agents [2].

7. REFERENCES

[1] Amazon website. http://www.amazon.com.

[2] eBay website. http://www.ebay.com.

[3] R. R. Bouckaert, E. Frank, M. Hall, R. Kirkby, P. Reutemann, A. Seewald, and D. Scuse. *Weka manual (3.7.1)*, June 2009.

[4] A. Frank and A. Asuncion. UCI machine learning repository, car evaluation data set, 2010.

[5] T. Huynh, N. R. Jennings, and N. Shadbolt. An integrated trust and reputation model for open multi-agent systems. *Journal of Autonomous Agents and Multi-Agent Systems*, 13(2):119–154, 2006.

[6] G. Lu, J. Lu, S. Yao, and J. Yip. A review on computational trust models for multi-agent systems. In *International Conference on Internet Computing*, pp. 325–331. CSREA Press, 2007.

[7] T. M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997.

[8] J. Sabater and C. Sierra. Regret: reputation in gregarious societies. *Procs AAMAS'01*, pp. 194–195, 2001.

[9] E. Serrano, A. Quirin, J. A. Botía, and O. Cordón. Debugging complex software systems by means of pathfinder networks. *Information Science*, 180(5):561–583, 2010.

[10] E. Serrano, M. Rovatsos, and J. Botia. Mining qualitative context models from multiagent interactions (extended abstract). *Procs AAMAS'11*, pp. 1215–1216, 2011.

[11] W. T. L. Teacy, J. Patel, N. R. Jennings, and M. Luck. Coping with inaccurate reputation sources: experimental analysis of a probabilistic trust model. *Procs AAMAS'05*, pp. 997–1004, 2005.

[12] B. Yu and M. P. Singh. An evidential model of distributed reputation management. *Procs AAMAS '02*, pp. 294–301, 2002.

[13] G. Zacharia and P. Maes. Trust management through reputation mechanisms. *Applied Artificial Intelligence*, 14(9):881–907, 2000.

[2]Acknowledgments: This research work is supported by the Spanish Ministry of Science and Innovation under the grant AP2007-04080 and in the scope of the Research Projects TSI-020302-2010-129, TIN2011-28335-C02-02 and through the Fundación Séneca within the Program 04552/GERM/06.

PRep: A Probabilistic Reputation Model for Biased Societies

Yasaman Haghpanah
University of Maryland Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250
yasamanhj@umbc.edu

Marie desJardins
University of Maryland Baltimore County
1000 Hilltop Circle, Baltimore, MD 21250
mariedj@umbc.edu

ABSTRACT

Several reputation models have been introduced to deal with the problem of biased reputation providers. Most of these models discount or discard biased information received from the reputation providers, and most of them are not appropriate when a large population of information providers are biased or dishonest. In this paper, we present a probabilistic approach for reputation modeling, the Probabilistic Reputation model (PRep). PRep models a reputation provider's behavior, and uses this model to re-interpret the reported information, thus making use of the entire reputation reports effectively, even if they are biased. The re-interpreted data is combined with the agent's direct experiences to determine an overall level of trust in the third-party agent. We show that PRep significantly outperforms two state-of-the-art trust and reputation models—HAPTIC and TRAVOS—and improves the overall payoff in a game-theoretic environment.

Categories and Subject Descriptors

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

General Terms

Human Factors, Design, Experimentation

Keywords

Reputation, Trust, Bayesian learning, Behavioral modeling

1. INTRODUCTION

Researchers have used reputation to model the trustworthiness of individuals in online markets, such as eBay, Amazon, and Yahoo [2, 4, 7]. eBay's tremendous success as an online auction site stems largely from its powerful yet simple reputation system, Feedback Forum [7]. The importance of reputation systems in Internet-mediated service provision has been widely recognized by researchers in various disciplines, such as multi-agent systems, economics, and information systems [4].

In the literature, reputation has been referred mostly to the aggregation of people's opinion about one person. In this paper, we use reputation as the perception of one person (or agent) about another person's behavior, intention, or reliability of service. This perception depends directly on the reputation reporter's characteristics, such as their level of uncertainty, whether they are biased or realistic, and/or cultural biases they may have. Different reputation characteristics can be dominant in specific domains. For example, the buyers' behavior in eBay can be modeled as biased towards giving positive or negative reviews. In reality, eBay's feedback forum has been observed to be surprisingly positive: among all ratings provided in eBay's feedback forum, 99% are positive [2, 7].

Avoiding unfair ratings while obtaining unbiased and honest reviews and ratings has been shown to be problematic and extremely hard to achieve [2]. Researchers in this area have explored different solutions to this problem. Some have tried to solve it by identifying unbiased reviews and using models that discount or discard the biased information [10, 12, 13]. Another proposed approach is to define a measure of review helpfulness, and identify the helpful reviews among a set of candidate reviews [3]. These approaches help to reduce the effect of biased and non-realistic reviews, and therefore highlight unbiased information that can be used for decision making. However, these proposed models are also throwing away data by filtering, discounting, and discarding, despite the fact that reviews are costly and in general not easily obtainable. Additionally, some products have few reviews, providing too little data to identify the fair reviews and discount the rest [3].

We propose the Probabilistic Reputation (PRep) model, a novel solution grounded in probabilistic modeling that learns the reviewers' behavior using Bayesian learning and then adjusts their reviews or ratings, as opposed to finding the unbiased reviews and discarding the rest. In the PRep framework, an agent first gathers information about a target agent through both direct interactions with that target and a reviewer's report about the target. Then, it learns the reporting agent's behavior by comparing these two sources (i.e., reports and direct experiences). After the learning phase is complete, the PRep agent can interpret other reports about other targets coming from the same report provider. As a result of this interpretation, it uses the entire report effectively, even if the report provider is biased (i.e., even if its reports are based on faulty perceptions or on dishonest reporting).

The key benefits of PRep are:

- The PRep reputation mechanism uses biased information as well as unbiased information; it therefore benefits from all available data.
- PRep agents obtain a tailored view of the reviewer (or reporter) according to their own behavior and preferences, resulting in customized aggregation of reviews.
- PRep is still effective in cases with very few observations or reviews. Most current models are unable to find usable feedback or generate a meaningful reputation level when only a few ratings are available [3].

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

In this paper, we describe our approach and its application in a game-theoretic environment. Our experimental results show that PRep is able to learn the reporting behavior of a report provider, and consequently to interpret other reports of that provider, resulting in better decision making and higher payoffs in its future interactions. Also, our results show that PRep identifies other agents' trustworthiness faster and more accurately than two other state-of-the-art trust and reputation models (TRAVOS and HAPTIC), even when reported information is biased.

2. RELATED WORK

Reputation has been widely studied [2, 4, 7, 8]. Several reputation models and mechanisms have been proposed in the literature to deal with the problem of biased and unfair ratings.

The BRS [13] and TRAVOS [10] approaches construct Bayesian models, using the number of satisfactory and unsatisfactory interactions with the sellers as ratings, and then use outlier detection or relevance analysis to filter out unreliable ratings. A drawback of these approaches is that a significant amount of information may be considered unreliable, and therefore discarded or discounted. BLADE [6] uses a Bayesian model reputation framework. In contrast to BRS and TRAVOS, it does not discard all unreliable ratings; rather, it learns an evaluation function for advisors who provide ratings close to their direct experience. Therefore, BLADE only performs well if the advisors are extremely honest or extremely dishonest. For example, BLADE discounts the ratings even if the advisor provides 70% honest reports. In the real world, advisors are not purely good or bad and could have various levels of honesty.

Vogiatzis et al. [11] proposed a probabilistic trust and reputation model that focuses on modeling service providers whose behavior is not static with time. Their model does not work well in the presence of biased advisors. Additionally, Vogiatzis's model and TRAVOS both assume that there has been a history of interactions between the agent (i.e, the reputation requester) and a service provider. Noorian et al. [5] categorize an advisor's "unfairness" behavior into two groups: intentional and unintentional. Their model, Prob-Cog, uses a two-layer filtering approach to detect and disqualify unfair advisors. Prob-Cog mainly targets and filters out advisors who are intentionally biased. Their model does not perform well when there is a large population of intentionally unfair advisors.

Zhang and Cohen [15] proposed a personalized approach to handle unfair ratings. They use private and public reputation information to evaluate the trustworthiness of advisors. They estimate the credibility of advisors using a time window to calculate the recency of ratings, and then estimate the trustworthiness of advisors based on the ratings. Their model does not interpret unfair ratings. As a result, when the proportion of unfair ratings increases, the trustworthiness of advisors decreases; this results in the system relying heavily on private reputation (i.e., agent's direct experiences). Yu and Singh [14] measure how much the advisor's rating deviates from the consumer's experience. Their model identifies accurate advisors, and discards deceptive advisors.

Another area of research is focused on sentiment analysis and review helpfulness. For example, Kim et. al. [3] propose a method for automatically determining the quality of reviews. They use regression to rank different sets of reviews on Amazon.com, based on their helpfulness. They do not customize the reviews based on a user's experiences or preferences. Also, since many products receive very few reviews, their approach is not helpful for such cases.

In contrast to these mentioned models, PRep uses and customizes reviews (or reports) even when they are biased. Without prior interactions with a service provider, a PRep agent can form a view about the service provider by requesting and interpreting the opinion of an advisor, if it has previously observed the advisor's behavior. This allows PRep agents to form a view about service providers that have very few reviews and ratings or for whom the majority of the reviews is biased. Other reputation models do not work effectively in such cases.

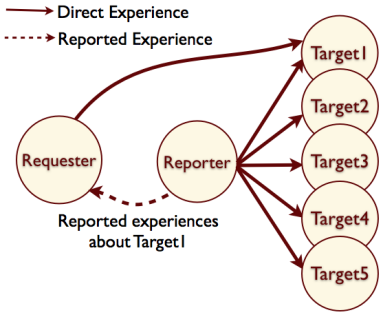


Figure 1: Basic scenario. Requester stands for Reputation Requester, Reporter stands for Reputation Reporter, and Targets are agents that Requester would like to know about.

3. THE PREP MODEL

In this section, we explain our reputation mechanism, PRep, which is based on probabilistic modeling and Bayesian learning. PRep has two main steps: learning the reporter's behavior (Section 3.3) and interpreting the later reports coming from that reporter for use in decision making (Section 3.4).

Figure 1 explains our model using a two-step scenario involving a reputation *Requester*, a reputation (review or opinion) *Reporter* (advisor), and several *Target*s (service providers). In this scenario, Requester is new to a society of agents, but Reporter has been in this society for some time and has had direct interactions with several agents (Target1, Target2, Target3, etc.). Requester first starts to interact with Target1 directly, then asks Reporter for some information about Target1. By comparing its own direct experience to the reported experience of Target1, Requester learns Reporter's reporting behavior. At this point, Requester can interpret acquired reports from Reporter about other agents (e.g., Target2) and can use this information to interact more effectively with those agents. Note that Target1 does not know that Requester is new and has requested reputation information from Reporter. This assumption prevents Target1 from deliberately misleading the Reporter in order to mislead the Requester. Also, Reporter does not know whether Requester has already interacted with Target1. The latter assumption prevents Reporter from deliberately misleading Requester about its reporting behavior.

Trust and reputation have generally been modeled using two sources: direct and reported experiences. PRep interprets reported experiences in its reputation model and uses a direct-experience trust model to evaluate the trustworthiness of agents. In this paper, we use HAPTIC [9] as the trust model. However, PRep is general and can be combined with other existing direct-experience trust models.

3.1 Direct-Experience Trust Model

Harsanyi Agents Pursuing Trust in Integrity and Competence (HAPTIC), a trust-based decision framework, is among the few existing models with a strong theoretical basis: HAPTIC is grounded in game theory and probabilistic modeling. It has been shown that HAPTIC agents are able to learn other agents' behaviors reliably using direct experiences. One shortcoming of HAPTIC is that it does not support reported experiences.

The HAPTIC model allows an agent to predict a partner's actions and use these predictions to decide whether or not to trust that partner. The key insight in HAPTIC is that it separately models trust using two components of *competence* and *integrity*. Competence is modeled as the probability that a given agent will be able to execute an action in a particular situation. Integrity is an agent's attitude towards honoring its commitments (or equivalently as the agent's belief in a discount factor), and is affected by the perceived probability of future interactions. This distinction is useful when an agent defects. It is important for the other agent to understand whether the defection was due to the incompetence of an honest agent, or was the result of cheating by a competent agent with low integrity. HAPTIC identifies a discrete set of player types, denoted by Θ, and maps each agent's competence and integrity θ to a value from this set. A HAPTIC agent observes the behavior of other agents and estimates their competence and integrity, then uses this data for decision making in future interactions with each agent.

HAPTIC has been applied to a modified two-player Iterated Prisoner's Dilemma (IPD), in which the payoff matrix in each round is scaled using a random multiplier. As a result, the payoffs differ from one round to the next. HAPTIC assumes that agents know the current round's multiplier before selecting their actions. With variable payoffs, a failure due to low competence can be distinguished from a failure that results from low integrity. An honest but incompetent agent defects randomly, irrespective of the payoff. By contrast, a cheating agent shows a pattern in its defections that is correlated with the expected payoffs. A HAPTIC agent computes expected payoffs (as defined in the classic Prisoner's Dilemma payoff matrix) and decides rationally whether to cooperate or defect. Equation 1 defines δ, a threshold for cooperation and defection. If the agent's integrity is greater than δ, it will cooperate; otherwise, it will defect. δ can be computed for each agent and each game using the current round's payoff multiplier *m*, the average payoff *M*, and the estimates of the payoffs of the four possible outcomes ($\widehat{P}$, $\widehat{S}$, $\widehat{R}$, and $\widehat{T}$).[1]

$$\delta = \frac{1}{\frac{M(\widehat{P}-\widehat{R})}{m(\widehat{R}-\widehat{T})} + 1}. \quad (1)$$

A learning HAPTIC player considers the outcome of each round as either a Success (expected action) or a Failure (unexpected action), based on its hypothesis about that agent's type. Iterative games between two agents allow HAPTIC players to reduce the set of probable types being considered. The HAPTIC learning method uses observations of agent behavior to estimate the competence and integrity for each agent.

3.2 Types of Reporters

One of the dominant recognized reviewer behaviors (including eBay's Feedback Forum) is being positively or negatively biased. In the real world, some reviewers are realistic (and honest), truthfully providing the requested information, reviews, or rates. Others tend to hide people's defects because they are afraid of retaliation [7], they are hopeful of getting a good rate in return [1], or they gain personal or economic rewards or incentives by doing so. Still others may change the results with pessimism, because they are pessimistic people by nature, or because they want to ruin a competitor's reputation and discredit them. Note that reporting negatively about a service can be completely realistic and not pessimistic, if the service was actually bad.

To address the consequences of these behaviors in the real world, we model the behavior of reporters in PRep as being potentially biased. We define the reporters' behavior using three types: *realistic*, *optimistic*, and *pessimistic*, similar to Noorian et al.'s approach

[1]R, T, S, and P are the standard PD payoffs from the payoff matrix.

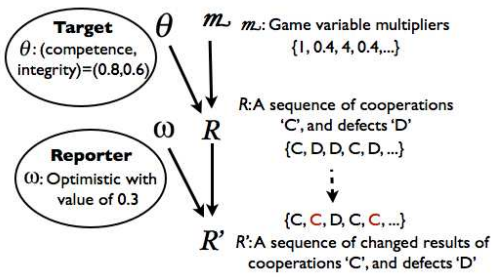


Figure 2: Report generation from a game between Reporter and Target.

[5]. A realistic Reporter always reports truthful information, corresponding directly to the experiences that it has had in the past with other agents. A pessimistic Reporter underestimates other agents' behavior, and an optimistic Reporter overestimates other agents' behavior. The level of optimism (or pessimism) is modeled by an ordered pair, $\omega = (\omega_{opt}, \omega_{pess})$, which may be based on the Reporter's innate characteristic or could depend on Reporter's incentives for honesty/dishonesty. Specifically, with probability ω_{opt}, Reporter will change some of the Defect actions of the target into Cooperates in its reports. Similarly, ω_{pess} defines the probability of changing Cooperate actions into Defects. For optimistic reporters, ω_{opt} represents the degree of optimism (probability of a $D \rightarrow C$ "flip"), and ω_{pess} is zero. Likewise, for pessimistic reporters, ω_{pess} is the degree of pessimism, and ω_{opt} is zero.

Figure 2 shows how a report is generated in an IPD environment, and how it will be changed by different reporters. We denote the actual result of the series of games between Reporter and Target as R. R is a sequence of Cooperate and Defect actions by Target in the series of games played with Reporter. The interactions and reporting process are as follows. Target makes its decisions based on its competence and integrity, θ, and the payoff multiplier *m* of each game, as modeled in HAPTIC. When Reporter wants to submit R to Requester, it will first change R to R' based on its type, ω, and then deliver R' to Requester. For example, if ω is 30% optimistic, then Reporter will change each Defect (in R) to a Cooperate (in R') with probability 0.3 (Figure 2).

In the real world, a Reporter could have various perceptions of interacting with different targets, based on its relationship with those targets, e.g., as a collaborator or competitor. Here, however, we assume that Reporter has the same perception of plays with different Targets, so its reporting behavior will be the same for various Targets. Since HAPTIC assumes that agents know the current multiplier of the round, we maintain this assumption here: all agents know the multipliers of the games. We intend to relax both of these assumptions in our future work.

3.3 Learn Reporter's Type

We now explain how Requester learns Reporter's type using Bayesian model averaging by comparing direct and reported experiences. Consider our basic scenario, shown in Figure 1, in which Requester and Reporter have played separately with Target1. Suppose that Requester asks Reporter for some information about Target1. We denote the actual results of the play between Reporter and Target1 by R, and between Requester and Target1 by D. Reporter changes the true results, R, based on its type, ω, to R' for reporting to Requester.

We define a set of discrete reporter types, Ω.[2] Each type $\omega_i \in \Omega$

[2]Using a discrete set of possible agent types is simpler and less

is a pair of values (ω_{opt}, ω_{pess}). Realistic agents are modeled by ω = (0, 0). The probability of a type hypothesis ω_i is denoted by $P(\omega_i)$. Requester has also learned a probability distribution over the possible player types for Target1, which are denoted by θ_j. The probability of each player type is denoted by $P(\theta_j)$.

To find the probability of each type of Reporter, given the results R' and D, i.e., $P(\omega_i|R', D)$ for each Reporter type, ω_i, we use Bayesian model averaging over all possible Target1 types, θ_j:

$$P(\omega_i|R', D) = \sum_{\theta_j \in \Theta} P(\omega_i|R', D, \theta_j) \times P(\theta_j|R', D). \quad (2)$$

The second term, $P(\theta_j|R', D)$, is the probability of Target1's type being θ_j, given R' and D. In this case, D, the direct experience, is more reliable than R', the reported experience. Therefore, PRep conditions θ_j only on D, and this term is simplified as $P(\theta_j|D)$, which is Requester's probability distribution of Target1's type, learned using the HAPTIC model.

The first term, $P(\omega_i|R', D, \theta_j)$, is the probability of a Reporter's type, given Target1's type θ_j, R', and D. Since ω_i is conditionally independent of the results of Requester and Target1's play (D) given θ_j and R', this term can be simplified to $P(\omega_i|R', \theta_j)$. Using Bayes's rule, we can rewrite this term as:

$$P(\omega_i|R', \theta_j) = \frac{P(R', \theta_j|\omega_i) \times P(\omega_i)}{P(R', \theta_j)}. \quad (3)$$

We assume a uniform prior on the Reporter's type, so $P(\omega_i)$ is just the reciprocal of the number of defined types for Reporter ($P(\omega_i) = \frac{1}{|\Omega|}$). Also, $P(R', \theta_j)$ is a normalizing factor, so we only need to compute $P(R', \theta_j|\omega_i)$. Using the definition of conditional probability, this term can be rewritten as:

$$P(R', \theta_j|\omega_i) = P(R'|\theta_j, \omega_i) \times P(\theta_j|\omega_i). \quad (4)$$

Since θ_j and ω_i are independent, the second term in Equation 4 is $P(\theta_j)$, a prior uniform distribution over the player types. The expected value of $P(R'|\theta_j, \omega_i)$ is defined by a weighted sum over all possible values of R:

$$E(P(R'|\theta_j, \omega_i)) = \sum_R P(R'|R, \theta_j, \omega_i) \times P(R|\theta_j, \omega_i). \quad (5)$$

Since computing this full expectation is computationally very expensive, one can instead approximate $P(R'|\theta_j, \omega_i)$ using the maximum likelihood value for R. Denoting the most likely R as R^*, this maximum likelihood can be written and expanded as:

$$P(R'|R^*, \theta_j, \omega_i) = P(R'_C, R'_D|R^*, \theta_j, \omega_i), \quad (6)$$

where R'_C are all the cooperates and R'_D are all the defects in the report. Since each round played is assumed to be independent of the others, the probabilities of the observed defects and cooperates in the report are independent of each other, yielding:

$$P(R'_C, R'_D|R^*, \theta_j, \omega_i) = P(R'_C|R^*, \theta_j, \omega_i) \times P(R'_D|R^*, \theta_j, \omega_i). \quad (7)$$

Each term in Equation 7 represents a series of i.i.d. (independent and identically distributed) observations from a Bernoulli distribution, so a binomial distribution can be used to compute the overall probability of each reporter type. The first binomial is the probability of observing a certain number of optimistic flips (i.e., the case where the intention R^* of Target1 is Defect and the report of that round, R', is Cooperate). The second binomial likelihood is the probability of seeing the observed number of pessimistic flips in the report. (when the intention R^* is Cooperate, but is reported as a Defect in R'). The expected success rate for the first binomial is the number of $D \rightarrow C$ flips over total number of Cooperates in the results, R'_C, that would be expected from a reporter with type ω_i. Similarly, the expected success rate for the second binomial is the number of $C \rightarrow D$ flips over R'_D. Note that a success in this context is a "flip": that is, when Reporter changes a Cooperate to a Defect, or vice versa. We multiply these two binomial likelihoods to compute $P(R'|\omega_i, \theta_j)$ in Equation 7. By averaging over all possible Target1 types, Requester can calculate the probability of each type of Reporter (Equation 2).

In more complicated environments, the Requester may have multiple reports from the same Reporter. In this case, we first learn the Reporter's behavior in each set of reports, and then use a weighted averaging function over all possible Reporter types, i.e., for N reports. In fact, to estimate the credibility of the learned ω in each transaction, we use the length of each report, i.e., the number of rounds for which two agents interacted with each other in each run:

$$P(\omega_i|R'_1, D_1, .., R'_N, D_N) = \frac{\sum_{j=1}^{N} P(\omega_i|R'_j, D_j) \times length(R'_j)}{\sum_{k=1}^{N} length(R'_k)}, \quad (8)$$

where $length(R'_j)$ is the number of interactions reported in R_j. Note that as the number of rounds increases, the statistics become more accurate, leading to better results (see Section 4).

3.4 Report Interpretation

In the previous subsection, Requester learned Reporter's type. In this section, the maximum likelihood of the possible Reporter types (i.e., $P(\omega_i|R', D)$) will be used to interpret the reported results for new Targets. We illustrate how agents can use this interpretation to learn the player types (competence and integrity) of other targets with whom they have not previously interacted.

After learning Reporter's type, Requester asks Reporter for information about Target2, and uses its learned knowledge of Reporter's type to interpret the reported results (which are denoted by R'_2). Without loss of generality, we explain how to interpret the reports when Reporter's type is optimistic. Recall that ω_{opt} represents the probability of optimistic flips in the report and ω_{pess} represents the probability of pessimistic flips in the report. Using Equation 9, an "interpret" function estimates the total number of Cooperates ($count_{R_{2C}}$ in the actual results R_2) using $count_{R'_{2C}}$, as the total number of reported Cooperates in the sequence R'_2, $length(R_2)$ as the number of rounds in the play, and ω_{opt}. The difference between $count_{R_{2C}}$ and $count_{R'_{2C}}$ is the number of Cooperates that should be changed back to Defects to produce more accurate results, and saving the result as R^*_2.

$$count_{R'_{2C}} = count_{R_{2C}} + \omega_{i\_opt} \times (length(R_2) - count_{R_{2C}}). \quad (9)$$

Requester now plays back the new results, R^*_2—generating an action as it would do if it were actually playing with Target2—and uses HAPTIC to update $P(\theta_j)$ for each possible Target2 player type, $\theta_j = (C, I)$. This distribution will continue to be updated in the online learning process between Requester and Target2, when they start their direct interactions. This knowledge will increase Requester's overall and mean payoff.

4. EXPERIMENTS

In this section, we present our experimental results. We show the performance of the learning and report interpretation components of PRep. We also compare the overall performance of PRep, HAPTIC, and TRAVOS in terms of learning accuracy and payoffs.

computationally expensive than modeling agent types with a continuous variable. We experimented with a continuous version, and the results are very close to what we obtain with discrete sets.

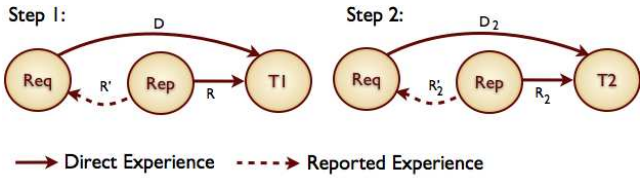


Figure 3: Step1 and Step2 of basic scenario. Req is Requester; Rep is Reporter; T1 & T2 are Targets.

As an overview, in the first two experiments, Exp1 and Exp2, we evaluate PRep's learning and interpretation components, respectively. In the third experiment, Exp3, we compare PRep with HAPTIC, and verify the results with a t-test. Finally, in Exp4, a TRAVOS Requester competes with a PRep Requester in finding Target1's behavior. We compare their mean error in finding Target1's behavior and the mean and cumulative game payoffs.

4.1 Simulation Parameters

Distribution of Reporter Types: In these experiments, the reporter type is chosen randomly using either a uniform distribution or a capped Gaussian distribution. These functions randomly generate numbers in the range (-0.7, 0.7), based on the type of distribution. A negative number represents a pessimistic reporter; a positive number is an optimistic reporter; and zero is realistic. We define the Gaussian distribution function with zero mean and a specified variance. Various demographics of realistic, pessimistic, and optimistic agents will be achieved by changing the variance of the Gaussian function.

PRep represents the set of possible reporters using a discrete set of types (ω_{opt}, ω_{pess}). Fifteen reporter types are considered by PRep: (0.1, 0), (0.2, 0)..(0.7,0) as optimistic reporter types; (0, 0.1), (0, 0.2),..(0,0.7) as pessimistic reporter types; and (0, 0) as a realistic reporter type. The uncertainty associated with the reporter's type is described by a multinomial probability distribution over these possible types. Learning of ω occurs by updating this probability distribution based on the observed behavior of that reporter after each reporting interaction.

Agent Strategies: Requester and Reporter are HAPTIC agents that have competence and integrity.[3] Targets are selected from classic strategies from the IPD literature in our experiments: ALLC, ALLD, TFT, and TFTT. An ALLC Target always cooperates in its play with any opponent. An ALLD Target always defects. A TFT (Tit-for-Tat) initially cooperates and then copies its counterpart's action from the previous round. A TFTT (Tit-for-Two-Tat) agent is forgiving and defects only if the opponent agent has defected twice in a row. We also use two variable-payoff strategies from Smith and desJardins [9]: DHP (Defect on High Payoff) and DMP (Defect on Medium Payoff). A DHP Target defects only on high-payoff games, and a DMP defects on medium and high payoffs, and cooperates on low payoffs.[4] Among these strategies, TFT and TFTT are the only ones who behave in reaction to their opponent's actions. The rest select their actions based on their type and regardless of their opponent's move.

We also introduce a noise factor for each of these strategic types, corresponding to HAPTIC's notion of competence. This factor, which is the probability of the actual action to be equal to the intended action, is selected from this set: {0.7, 0.8, 0.9, 1}.

[3] As in Smith and desJardins, competence of agents are selected from {0.7, 0.8, 0.9, 1}; and integrity is a number from 0 to 1.

[4] Multipliers of the rounds are selected from {0.4,1,4}. A DHP defects on rounds with m=4 and DMP defects on m=1 and 0.4 [9].

4.2 Exp1: PRep Learning

In our first experiment, we show the performance of PRep's learning component for different reporter types. We compare the given Reporter type distribution with the learned distribution and measure the accuracy of the learned Reporter types.

Design: We evaluate PRep in two steps, shown in Figure 3 (which follows our basic scenario presented in Figure 1). In the first step, PRep learns ω; in the second step, it uses the learned ω to interpret the reports in its successive plays. In step one, Requester and Reporter each play 20 rounds with Target1. Then, Requester asks Reporter about its experience with Target1. Reporter converts the actual results, R, to R' based on its type ω, and passes the report, R', to Requester. Requester then learns the Reporter's type, ω, given R' and R (using the approach described in Section 3.3). In step two, Reporter plays 20 rounds with Target2 (results = R_2). Then, Requester asks Reporter about Target2. Reporter converts the actual results R_2 to R_2' based on its type ω, and passes the results to Requester. Requester interprets R_2' based on the learned ω, and generates R_2^*.[5] Requester plays back R_2^* and learns Target2's competence and integrity, denoted by (C, I). Finally, Requester plays for 20 rounds with Target2, starting with its learned values for Target2's (C, I).

In Exp1, 100 Reporter types, ω, are selected randomly from a uniform or Gaussian distribution. Requester and Reporter player types (Competence, and Integrity) values are (1, 0.9). Target1 and Target2 types are selected randomly from a set of 16 strategic types: namely, the cross products of 4 player types (ALLC, ALLD, DHP, and DMP) and 4 competence values (0.7, 0.8, 0.9, 1).

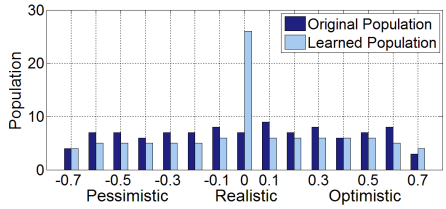
As a performance metric, we use the mean error, which is the difference between the identified Reporter type, ω, and the correct type. All results are averaged over 100 runs. [6]

Results: Figure 4(a) shows the distribution of true reporter types and most likely learned types in 100 runs of the experiment over 100 reporter types, when the true reporter types are selected using a uniform distribution. PRep is able to identify the uniform distribution, since the values are almost equally spread over the optimistic and pessimistic ranges, except for the realistic type (which will be explained next). The mean error for this experiment is 0.14. Part of this error arises from using discrete types in the learning process: the discrete steps are 0.1, so inherently an error up to 0.05 will be introduced during learning (0.025 on average).

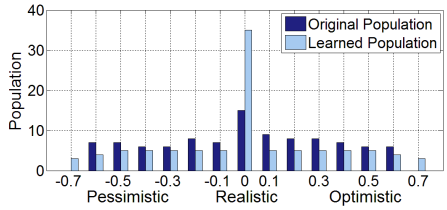
Another source of error is the population of learned realistic reporters (ω = 0), which is much higher (about 28) than the true number of realistic reporters value (100/15 or around 7). The explanation for this disparity is that optimistic reporters cannot be identified when they are reporting about ALLC players. An ALLC player always cooperates, so an optimistic reporter makes no changes in the report, and PRep detects such reporters as realistic. This problem can be solved when a PRep agent has multiple encounters with the same reporter (see Section 4.5). The same is true for pessimistic reporters when reporting about ALLD players. The population of ALLC players is 25, and roughly half those will face an optimistic reporter, which is 12 in the population. Similarly, another 12 false positives are generated from the ALLD players. Therefore, the population of realistic reporters will be estimated as 24 more than the true number. Since these misidentified realistic Reporters have a true value between 0.1 and 0.7, the average error for each of these

[5] R_2^* is Requester's estimation of what actually happened between Reporter and Target2, as R_2 is not available to Requester.

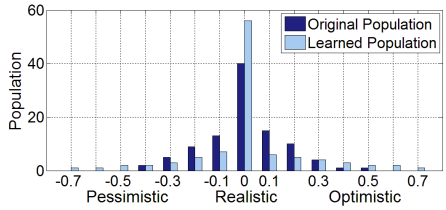
[6] Note that a "run" is different than a "round." A "round" is a single play between two agents in PD game, with single Cooperate or Defect as outcome. A "run" is a combination of several "rounds" in games between the agents in a scenario.



(a) Original distribution: uniform



(b) Original distribution: Gaussian with variance 0.3



(c) Original distribution: Gaussian with variance 0.1

Figure 4: Exp1; The probability associated with Reporter's true reporting type.

24 Reporters will be 0.4. This will cause an additional 0.096 (i.e., 0.4×24/100) error, making the estimated overall error to be 0.121 (0.025 + 0.096), which is very close to the actual error.

Figures 4(b) and 4(c) show the distribution of true agent types and most likely learned types over 100 reporter types, ω, selected from a Gaussian distribution, with variances of 0.3 (15% realistic) and 0.1 (41% realistic reporters), respectively. PRep is able to identify different distributions of reporters and the learned population is close to the original population for both large and small variances in the Gaussian function. The mean error for variance 0.3 is 0.11 and for variance 0.1 is 0.077. As the number of realistic reporters increases in the population, the mean error decreases; this occurs in part partially because fewer ALLC and ALLD targets will face optimistic or pessimistic reporters, respectively.

4.3 Exp2: PRep Interpretation

In the second experiment, our goal is to show the importance of correct interpretations when a reporter is biased. We design an experiment with fixed values (as a snapshot of Exp1), average it over 100 runs, and focus on finding the correct Reporter's type, ω and the Target's type, (C, I), after report interpretation and the resulting cumulative payoff.

Design: We follow the scenario shown in Figure 3. In the first step, Requester and Reporter play 30 rounds with Target1. In the second step, Requester and Reporter play 20 rounds with Target2.

In this experiment, we use HAPTIC as a baseline. Also, to show the negative effect of not re-interpreting reports, we define another baseline, PRep-NoInterp. This baseline uses PRep model without the interpretation component. A third baseline, PRep-Perfect, shows the upper limit benefit of reported experiences when the reporter is realistic and there are no flips in the report.

In Exp2, Reporter's true type is optimistic 0.4. Requester and Reporter player types values are fixed at (1, 0.9). Target1's (C, I) is: (1, 0.6), and Target2's true value is (0.7, 0.6).

Our performance metrics are the accuracy of the learned Reporter's ω and Target2's player types (by looking at the probability assigned to the true player types, i.e., (C, I)) and the cumulative payoff. The results are averaged over 100 runs.

Results: Figure 5(a) shows the results of learning Reporter's ω in Exp2, averaged over 100 runs, where ω is optimistic 0.4. This graph shows that Player1 was able to identify Reporter's type as having an optimistic behavior. The probability of the levels of optimism is spread over different values; the maximum likelihood of these values, is optimistic 0.4, with probability 0.22. This result illustrates the correctness of PRep's learning component.

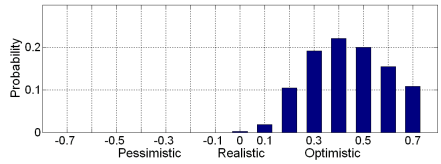
Figure 5(b) displays the results of learning Target2's (C, I). The possible hypotheses for Player4 are shown by small cross signs; the correct hypothesis is (0.7, 0.6), which is the true value of Target2 type. The circles' sizes represent the learned probability of each hypothesis for Target2. The top left graph shows the results for HAPTIC. In this case, Requester uses only direct experiences. After 20 rounds of play, the hypothesis probabilities are spread among four values: (0.7, 0.9), (0.7, 0.6), (0.7, 0.35), and (0.7, 0.1), which means that Requester is getting close but has not yet correctly identified Target2's true type. The PRep-NoInterp graph shows that using the non-interpreted reports still yields a moderate probability of finding the correct hypothesis. The results for PRep are shown in the bottom left graph, where the highest probability is assigned to (0.7, 0.6). This is the correct hypothesis; therefore, Requester can achieve higher payoffs with this learned model than using direct experience alone. If Reporter was a realistic reporter instead of being 40% optimistic in Exp2, Requester would have been able to identify Target2's actual (C, I) with a higher probability, as shown in PRep-Perfect graph in Figure 5(b).

Another interesting view of the learning process is how the learned probabilities changes over a series of rounds for Target2's true type. As seen in Figure 5(c), PRep starts high (near 0.56) from the beginning, while HAPTIC's probability of the true type remains at a lower level and needs several more rounds to increase. The main reason for this behavior is that PRep has learned Target2's type using reported experiences that it has received from Reporter.

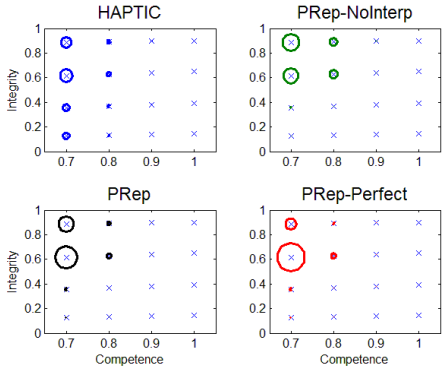
The corresponding payoffs resulting from the four approaches are shown in Figure 5(d). As expected, PRep-Perfect has the highest payoff; PRep (that interprets biased reports) ranks second and yields payoffs close to PRep-Perfect. HAPTIC places third; PRep-NoInterp is in the fourth place and behaves very similarly to HAPTIC. Since the reporter in this experiment alters Defects in the results with a 40% probability, using reports without interpretations will result in a performance close to HAPTIC, which is hindered by its belief in the incorrect reports.

4.4 Exp3: HAPTIC Vs. PRep

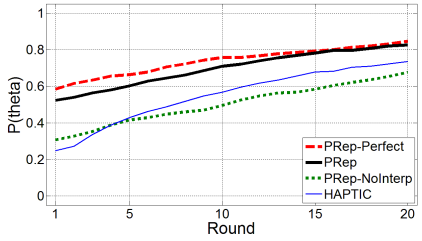
To verify the effectiveness of PRep over different player and reporter types, we performed Exp3, repeating a game for 100 times. In each run, we use the scenario in Figure 3. Requester's type is fixed at (1, 0.9), and the reporters' types are selected based on a Gaussian distribution with 0.3 variance (15% realistic reporters) centered on zero. The Target1 and Target2 types are selected randomly among 16 strategic types: the cross product of four strategic types (ALLC, ALLD, DHP, and DMP) with 4 competence values (0.7, 0.8, 0.9, 1). The mean payoffs for HAPTIC, PRep, and PRep-Perfect in this experiment are 1.89, 2.17, and 2.18, respectively. PRep (with biased reporters) achieves 14.8% improvement over HAPTIC, where the upper limit is 15.3% achieved by PRep-Perfect. A t-test confirms that the mean per-round payoffs of HAP-



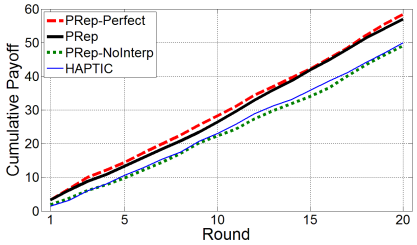
(a) Reporter's type probabilities for correct hypothesis of ω=(0.4,0)



(b) Target2's type (C, I) probabilities for correct hypothesis of (0.7,0.6)



(c) Target2's probability growth over rounds



(d) Cumulative payoffs

Figure 5: Exp2; A 40% optimistic Reporter and Target2 type actual values (C= 0.7, I = 0.6).

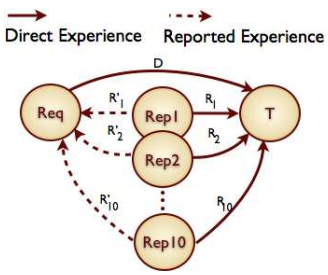


Figure 6: Scenario for TRAVOS and PRep. Req is Requester; R1, R2,..R10 are Reporters; and T is the Target.

TIC and PRep are different; with 99.9% confidence, the difference is between 0.274 and 0.276.

4.5 Exp4: TRAVOS Vs. PRep

In Exp4, we compare the performance of TRAVOS [10] and PRep in a noisy environment with biased and unbiased reporters. We measure the accuracy of the learned Target types, and the resulting mean and cumulative payoffs for both a PRep Requester and a TRAVOS Requester.

TRAVOS: This model uses probabilistic modeling based on a beta distribution. TRAVOS outperforms many other trust and reputation models, including probabilistic models such as BRS [13]. TRAVOS uses the number of satisfactory and unsatisfactory interactions with the sellers as ratings, and uses a weight function to combine these ratings. Agents calculate rating weights by comparing the relevance of each rating to their own opinions.

TRAVOS models the trustworthiness of each agent by a fulfillment factor, which is equivalent to "competence" in PRep. However, TRAVOS does not model the integrity of an agent. In order to compare PRep and TRAVOS, we settle this difference by providing the integrity of an agent as an input to TRAVOS, whereas PRep is searching in a two-dimensional space for competence and integrity. Note that this gives an advantage to TRAVOS.

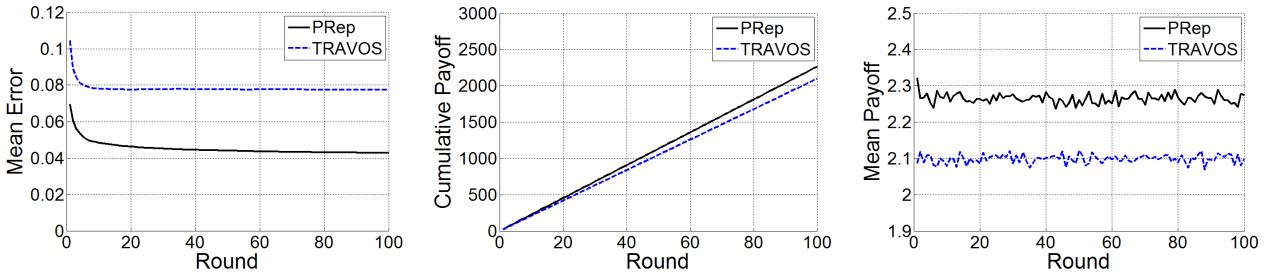
Design: To be able to compare PRep and TRAVOS in both modeling and assumptions, Exp4 uses another IPD-based test framework. TRAVOS assumes previous transactions between Requester and Target, so we design this experiment with this assumption. Also, we have several Reporters (each with different behavior) in this experiment reporting about one Target. Therefore, the Requester interprets different reporters' reports about one Target.

The scenario for this experiment is shown in Figure 6. Requester plays with Target for 10 rounds. Ten Reporters play 10 rounds with Target. Each Reporter changes the outcome of its play based on its type and then reports the changed results to Requester, who updates its belief about that specific Reporter. We repeat the same process 100 times; in each run, a Reporter's type, ω, is learned. In PRep, this value will be averaged over the so far learned ω (as seen in Equation 8) and later will be used in interpreting reports.

In this experiment, Requesters use either TRAVOS or PRep for modeling their trust and reputation; target types are selected randomly from the cross product of six strategic types (ALLC, ALLD, DHP, DMP, TFT and TFTT) with 4 competence values (0.7, 0.8, 0.9, 1). Requester and Reporter's competences and integrities are fixed at (0.8, 0.9). The population of Reporters consists of realistic and biased reporters (pessimistic/optimistic up to 0.7 and realistic), selected from a Gaussian distribution with 0.1 variance (41% realistic reporters) centered on zero.

We compare the accuracy of Target player types (competence in this experiment) learned by TRAVOS and PRep. As a performance metric, we use the mean error, which is the difference between the identified type and the correct type. Also, we compare PRep and TRAVOS in terms of the mean and cumulative payoff. All results are averaged over 100 runs.

Results: Despite the fact that we have provided TRAVOS with the correct integrity, as we can see in Figure 7(a), PRep outperforms TRAVOS in identifying the Target's type (competence). This error for TRAVOS has converged to 0.078 and for PRep to 0.043 (a 45% improvement over TRAVOS). The reason is that TRAVOS heavily discounts the biased reports, while PRep interprets and uses that data to learn more about the behavior of the Target. As a result of correctly identifying the behavior of the Reporter, the cumulative payoff is increased from 2085 to 2264 (Figure 7(b)) and the average payoff per round is increased from 2.09 to 2.26 (a 9% im-



(a) (b) (c)

Figure 7: Exp4; TRAVOS vs. PRep; (a) Mean error in identifying true Target's competence, (b) Cumulative, and (c) Mean payoffs for Requester in its play with Target.

provement), as shown in Figure 7(c). The results passed the t-test, which verifies the mean values of TRAVOS and PRep are different; with 99.9% confidence, the mean payoff difference is between 0.16 and 0.17.

We repeated this experiment with various number of rounds of direct experiences (i.e., "D" in Figure 6). The results show that TRAVOS is performing as well as PRep when the number of direct experiences is high. Figure 8 shows that the mean error of both models converges to the same value if we increase the number of direct interactions up to 30. This means that TRAVOS is heavily relying on direct experiences, and PRep is performing better when there are only a few direct interactions available. Additionally, it shows that mean error decreases for both TRAVOS and PRep when the number of realistic reporters increases in the population.

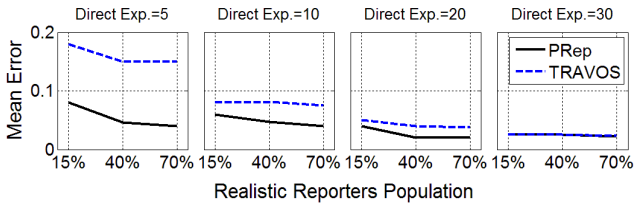


Figure 8: Exp4; Performance of TRAVOS vs. PRep using a variable number of direct experiences.

5. CONCLUSIONS AND FUTURE WORK

We presented PRep, a reputation mechanism that is capable of re-interpreting and adjusting reported experiences by learning the reporters' behavior. PRep works well in regular and noisy environments, even with the presence of a large population of biased reporters, and when there are only a few direct interactions available. Our results show that a PRep agent identifies agents' reporting behavior correctly; therefore, it recognizes other agents' trustworthiness more rapidly and accurately than a HAPTIC or TRAVOS agent, resulting in better decision making. For example, with 10 direct interactions, PRep's mean error for predicting an agent's behavior is 45% lower than that of TRAVOS, due to PRep's ability to correctly interpret the reports. As a result, the average payoff improves by 9%.

An interesting direction for future work would be to further evaluate this model in a risky and non-deterministic environment, such as a marketplace application. Also, we plan to explore the use of context-dependent Reporter types that can cause agents to behave differently in various situations (e.g., when reporting to a competitor versus a collaborator). We will also investigate multidimensional trust models that can be applied when a Reporter can have varying degrees of trust for different aspects of a Target's behavior (e.g., quality and price in a supply chain management context).

6. REFERENCES

[1] M. Chen and J. Singh. Computing and using reputations for internet ratings. In *Proceedings of the 3rd ACM Conference on Electronic Commerce*, pages 154–162, 2001.

[2] A. Josang, R. Ismail, and C. Boyd. A survey of trust and reputation systems for online service provision. *Decision Support Systems*, 43(2):618–644, 2007.

[3] S. Kim, P. Pantel, T. Chklovski, and M. Pennacchiotti. Automatically assessing review helpfulness. In *EMNLP-06*, pages 423–430, 2006.

[4] L. Mui, M. Mohtashemi, and A. Halberstadt. Notions of reputation in multi-agent systems: A review. In *AAMAS-02*, pages 280–287, Bologna, Italy, July 2002.

[5] Z. Noorian, S. Marsh, and M. Fleming. Multi-layer cognitive filtering by behavioral modeling. In *AAMAS 2011*, pages 2–6, Taipei, Taiwan, 2011.

[6] K. Regan, P. Poupart, and R. Cohen. Bayesian reputation modeling in e-marketplaces sensitive to subjectivity, deception and change. In *AAAI-06*, volume 21, page 1206. AAAI Press, 2006.

[7] P. Resnick, K. Kuwabara, R. Zeckhauser, and E. Friedman. Reputation systems. *Communications of the ACM*, 43(12):45–48, 2000.

[8] J. Sabater and C. Sierra. Review on computational trust and reputation models. *JAIR*, 24(1):33–60, 2005.

[9] M. Smith and M. desJardins. Learning to trust in the competence and commitment of agents. *Journal of AAMAS*, 18(1):36–82, February 2009.

[10] W. Teacy, J. Patel, N. Jennings, and M. Luck. Coping with inaccurate reputation sources: Experimental analysis of a probabilistic trust model. In *AAMAS-05*, pages 25–29, 2005.

[11] G. Vogiatzis, I. MacGillivray, and M. Chli. A probabilistic model for trust and reputation. In *AAMAS-10*, pages 225–232, 2010.

[12] Y. Wang, C. Hang, and M. Singh. A probabilistic approach for maintaining trust based on evidence. *JAIR*, 40:221–267, 2011.

[13] A. Whitby, A. Jøsang, and J. Indulska. Filtering out unfair ratings in Bayesian reputation systems. In *Proceedings of the 7th Int. Workshop on Trust in Agent Societies*, 2004.

[14] B. Yu and M. Singh. Detecting deception in reputation management. In *AAMAS-03*, pages 73–80, 2003.

[15] J. Zhang and R. Cohen. Evaluating the trustworthiness of advice about seller agents in e-marketplaces: A personalized approach. *Electronic Commerce Research and Applications*, 7(3):330–340, 2008.

A Decision-Theoretic Characterization of Organizational Influences

Jason Sleight and Edmund H. Durfee
Computer Science and Engineering
University of Michigan
Ann Arbor, MI 48109
{jsleight,durfee}@umich.edu

ABSTRACT

Despite a large body of research on integrating organizational concepts into cooperative multiagent systems, a formal understanding of how organizations can influence agents' decisions remains elusive. This paper works toward such an understanding by beginning with a model of agent decision making based on decision-theoretic principles, and then examining the possible routes that organizational influences can take to affect that model. We show that alternative avenues of applying influences correspond to different prior notions of organizational control, and empirically demonstrate the impact that each can have on the quality and overhead of coordinated behavior. To do so, we must define the agents' baseline behavior (without a designed organization), and we present a methodology for initializing agents' models to comprise what amounts to an "uninformed" organization. Finally, we show how the specification of organizational influences in terms of components of a decision-theoretic agent creates opportunities for agents to compare actual events with predictions implied in the models, such that agents can reason about whether to change organizations. We demonstrate that this capability to question and change organizations can be valuable if used judiciously.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Coherence & co-ordination, multiagent systems*

General Terms

Design, performance

Keywords

Organization, organizationally adept agents

1. INTRODUCTION

Organizational structuring is a widely adopted and often powerful tool for coordinating large groups of people to achieve common goals effectively and efficiently, by giving each person guidance in how to make local decisions that are useful to the collective endeavor. Multiagent systems research has investigated how organizational concepts and strategies can be modeled and utilized by computational agents, showing that organizations can increase the expected performance of large-scale, cooperative multiagent systems [12, 6]. Research also suggests that organizational control becomes increasingly effective as the number of agents increases, the time horizon increases, the system complexity increases, the system resources decrease, and/or the performance goals increase [4]. That these issues arise in realistic application domains has driven research into how to encode pertinent organizational control and how to augment agent architectures to follow such control.

A point of departure in this paper is that we attack the question of what an organization is or could be, computationally, from the opposite direction. We begin with a model of agent decision making based on decision-theoretic principles, captured as decentralized partially observable Markov decision processes, Dec-POMDPs. Within this formal, well-defined decision framework, we then explore how various types of organizational control and influences can be captured in the different components of the framework, such as transition and reward functions. Hence, one contribution of this paper is a systematic and comprehensive enumeration of where organizational control can be applied, and how it can be formally manifested in decision-theoretic agents. We empirically evaluate how the embodiment of organizational influence in different Dec-POMDP framework components individually and collectively impacts the quality of agents' behavior and the costs of agents' reasoning.

Measuring performance improvements resulting from designing and following a good organization requires a baseline of performance without any organization. Our more principled formulation reveals, however, that defining such a baseline is problematic. A second contribution of this paper, therefore, is a methodology for forming baseline organizations for experimental comparisons.

Our third main contribution in this paper is to demonstrate how an explicit representation of organizational control in terms of components of a decision-theoretic framework captures statistical predictions about runtime behavior, which agents can use to decide how and when to change (or abandon) their organization. We build off of the abstract concept of an organizationally adept agent (OAA) [3] to formulate a more precise notion of an OAA that can compare actual experiences in its environment with the organization's predictions, and can (with other OAAs) adopt a better alternative organizational design. Our preliminary experiments show that this capability to question an organization's suitability

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

rather than to follow it blindly can improve system-wide performance, but that responsiveness needs to be tempered by the costs of reorganization.

The remainder of this paper is structured as follows. In Section 2, we describe the decision-theoretic framework that our agents use and that, for the purposes of this paper, the organizational structure must work within. Then we turn to our first contribution, in Section 3, where we describe how organizational influence is manifested in each of the components and how the different manifestations capture prior organizational strategies in the literature. In Section 4 we make our second contribution by analyzing the space of baseline organizations to consider and justifying our choices for our experiments. Section 5 presents our empirical evaluation of the impact of different forms of organizational influence on the quality and costs of coordination. We then turn to a description of how a rudimentary form of organizational adeptness has been captured in our agents and present preliminary experiments illustrating the promise and potential costs of agents that can change organizations (Section 6). We conclude in Section 7 with a summary of the work presented here and of our ongoing efforts.

2. PROBLEM REPRESENTATION

We adopt a standard Dec-POMDP decision model [2], $\mathcal{M} = \langle \mathcal{N}, S, \alpha, A, R, P, \Omega, O, T \rangle$, where: $\mathcal{N}$ is the set of n cooperative agents; S is the (finite) set of global states; α is a probability distribution over initial global states; A is the (finite) set of possible joint actions; R is the joint reward function; P is the joint transition function; Ω is the (finite) set of possible joint observations; O is the joint observation function; and T is the finite time horizon. Given a full specification of the Dec-POMDP, an optimal joint policy, π^*, can be formulated in principle. In practice, however, finding such a policy for anything but very simple problems (with few agents and small state and action spaces) is intractable [2], and even if found, executing such a policy is problematic because it generally assumes that all agents have the same beliefs about the global state.

For these reasons, multiagent approaches to solving such problems often assume that each agent possesses a local view of the joint problem. As is customary in that work, we assume that state is factored: every state is represented using the same set of τ state features, such that $\forall s \in S, s = \langle f_1 \in F_1, \cdots, f_\tau \in F_\tau \rangle$, where F_j is the finite set of possible values for state feature j. Each agent i has a local state representation S_i consisting of a subset of the τ features. Agent i has a local decision model defined for this state space: $\mathcal{M}_i = \langle S_i, \alpha_i, A_i, R_i, P_i, \Omega_i, O_i, T_i \rangle$, where local rewards, transitions, actions, etc. are defined over the states in S_i. We further adopt the common assumption of local full observability (each agent i can exactly observe the values of all of its local state's features). Given these assumptions, the local decision model $\mathcal{M}_i$ of an agent i represents a local MDP, such that an agent can compute its (optimal) local policy π_i with respect to $\mathcal{M}_i$. The joint policy is then simply defined as $\pi = \langle \pi_1, \pi_2, ..., \pi_n \rangle$.

To illustrate a problem of this type, we use a simplified firefighting scenario, where firefighting agents and fires to be fought exist in a grid world (Figure 1). The global state consists of the locations of the agents and the locations and intensities of the fires. Figure 1 shows an initial global state, where the locations of agents A1 and A2 are shown,

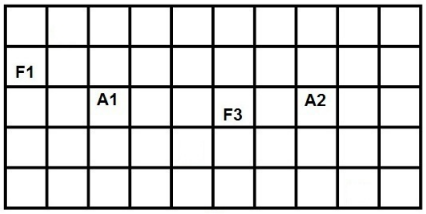


Figure 1: Example initial state of a 10×5 firefighting grid world domain. Ai indicates the position of agent i, and Fj indicates that there is a fire in that cell with intensity j.

along with positions of each fire Fx, where x is the current intensity of the fire in that position. Each agent has 6 actions: a NOOP action that makes no change to the world state; 4 possible movement actions (N, S, E, W) that move the agent one cell in the specified direction (and equates to a NOOP if there is no cell in that direction); and a fight-fire (FF) action that decrements by 1 the intensity of the fire in the agent's current location, if any and otherwise behaves like a NOOP. Joint actions are defined as the aggregation of the agents' local actions. Movement actions are independent (agents can occupy the same location), but FF actions are not: the intensity of a fire only decreases by 1 even if multiple agents simultaneously fight it. The joint reward for the agents in states prior to reaching T is the negative sum of the fires' intensities in that state. When the time horizon is reached, the problem episode ends, and the joint reward is 10 times the negative sum of the remaining fires' intensities, encouraging the agents to put all the fires out before the deadline.

An example of how agents might have local models of this joint model is the following. An agent's local state consists of its location and the locations and intensities of the fires. That is, it does *not* include the position of other agents. Hence, its local action space only includes its 6 actions, and its local transition model will only model how its local actions affect its local state. Its local reward function is the same as the global reward function; note that in this case the sum of the agents' local rewards will overestimate the true (negative) reward. Its local finite time horizon is identical to the global finite time horizon, and its local initial state distribution is calculated by directly mapping the initial distribution of global states into the local state space. Given such a local model, each agent will formulate a local policy that would fight the fires optimally if the agent were alone in the world. Note that, in general, the joint policy formed by the combination of these optimal local policies will not itself be optimal. For example, in Figure 1, both agents will be drawn to the high intensity fire first and redundantly fight it rather than dividing up to fight the two fires concurrently.

3. ORGANIZATIONAL INFLUENCE

As just illustrated, optimizing policies for local models of joint problems does not necessarily lead to optimal joint policies. Yet, as has been already discussed, centrally solving for an optimal joint policy is computationally infeasible and can lead to policies that rely on agents knowing the global state. The organizational approach that we examine here, therefore, focuses on modifying agents' local models such that the local policies that agents individually construct

will, in combination, result in better joint policies. We note that runtime communication to increase global awareness of agents' states and plans can also help improve coordination (e.g., help prevent firefighting agents from behaving redundantly), and could gainfully augment an organizational approach. However, in the remainder of this paper we assume no communication between agents in order to avoid confounding factors in our presentation and in our analysis of organizational influence's performance.

3.1 Organizational Design Space

We assert that the components of the Dec-POMDP model provide a way to systematically enumerate the dimensions of the organizational design space, at least for designs intended for decision-theoretic agents. Formally, let an organizational design be defined as $\Theta = \langle \theta_1, \cdots, \theta_n \rangle$ where $\theta_i = \langle S_{\theta_i}, \alpha_{\theta_i}, A_{\theta_i}, R_{\theta_i}, P_{\theta_i}, T_{\theta_i} \rangle$ is the local organizational model for agent i.[1] θ_i specifies the local state space, initial state distribution, action space, reward function, transition function, and finite time horizon (FTH) for agent i, when the agent is following organization Θ. We now step through each of the components and discuss how each could be used to introduce some commonly cited organizational influences.

Rewards: The idea of modifying local models to improve coordination is not new. In particular, a growing body of literature on *reward shaping* specifically looks at how agents' reward functions can be manipulated to bias agents into taking actions that benefit the collective [15, 10]. For example, reward shaping can lead an agent to establish conditions that have no (unshaped) local reward, but that enable other agents to then take actions that lead to high joint reward. In a similar spirit, Agogino and Tumer [1] have explored the process of designing agents' individual objective functions such that maximizing local rewards leads to maximizing a global objective function in expectation. Hence, one obvious dimension in the organizational design space is the space of alternative combinations of reward functions to assign to agents.

Transitions: It turns out, however, that changing each agent's local rewards alone might be insufficient to induce some forms of cooperative behavior. For example, consider the situation where one agent can establish a condition that enables another to take actions that ultimately lead to high reward. An organizational reward function can bias the first agent into establishing the condition; however, the second agent might not take useful precursor actions because its local model indicates that the condition is unlikely to be established by default. To induce the second agent into complementary behavior, the organizational designer needs to convey the expectation that, because of how the first agent's reward is shaped, the second agent should expect the condition to be (or become) established with high probability. The organization could give the second agent a modified transition function indicating that, given organizational influences elsewhere, the condition of interest is now more likely to be established. Note that the revised transition function summarizes the expectations without needing to be specific about the details; the second agent need not reason about how the first will establish the condition, or even which agent is establishing the condition.

[1]As mentioned, for simplicity we assume local state is fully observable. What follows can be extended to local partial observability with the usual impacts on complexity.

Hence, besides reward functions, transition function modification is another dimension of organizational design. While the example above points out how these can be correlated, even if agents' reward functions are left unchanged they could still benefit from improved transition functions, for example, by reflecting the tendencies that agents inherently have in affecting the states that others might face.

Actions: Without specialized optimizations during policy creation, organizational shaping of reward and/or transition components will not reduce the size of the agents' local policy spaces, but only their decisions about which of those policies are optimal. Redesigning some of the other components of an agent's decision model, however, can achieve another objective often attributed to organizational influence, which is to simplify an agent's reasoning. For example, the organizational designer might associate different roles with different agents and thus induce agents to specialize in the possible actions they will exercise. The designer can give agent i a reduced action specification $A_{\theta_i} \subseteq A_i$ that constrains its choices in some (or all) states. For example, in Figure 1, agent A1 might be prohibited from moving outside of an organizationally-dictated area of responsibility. Chosen well, such restrictions not only help agents pursue complementary policies, but simplify planning for each. Like reward shaping, encoding organizational influence as constraints on behavior is a familiar approach in the literature [6, 11].

States: In a factored state representation, the organizational designer could determine that there are features that an agent can sense that are unnecessary to represent given the organization. In our running firefighting example, for instance, the organizational designer might decide that some (distant) fires need not be modeled by an agent at all (because they are the responsibility of other agents), thereby simplifying its local decision problem. Further, the organizational designer might purposely augment an agent's local state representation with new features, where the designer has decided that those features are crucial to distinguishing between states that otherwise would look locally identical. Such augmentations must be done with caution, however, and if the designer includes such augmentations, it must also delineate the communication protocols and policies that would ensure an agent possesses up-to-date values for those features despite not being able to directly observe them. For instance, in our running example, to improve coordination the designer might insist that each firefighter tell the others which fire it is now working towards extinguishing. Establishing these types of commitments and conventions has proven useful [8], but this paper will only consider organizations that remove state features.

Initial State and FTH: Finally, an organization can also influence an agent's behavior through α_{θ_i} and T_{θ_i}. In the firefighting scenario, an organization could, for example, initially position the firefighters at particular locations and reflect the influence on initial state correspondingly. Similarly, by shaping the rewards, transitions, and actions of the various agents, the organizational designer might determine that the improved parallelism from coordination means that agents can safely reason over shorter time horizons. Alternatively, the designer might improve coordination by increasing T_{θ_i} for the agents, effectively asking them to be less myopic.

3.2 Related Work

In the preceding, we have stepped through the compo-

nents of a local decision model for a decision-theoretic agent, and described how an organizational designer could adjust a component to influence an agent's decisions. By adjusting the agents' components appropriately, an organizational designer can influence agents to make more complementary, globally-useful decisions, and in some cases also simplify the agents' local reasoning processes. As noted above, adjusting components like reward functions and action spaces have correspondences with familiar notions in the organizational structuring literature. However, prior work on implementing organizational influences within agents largely takes a top-down approach: given influences that a researcher's intuitions determine are pertinent, an agent architecture (such as a BDI architecture [3]) is extended to incorporate those influences. In contrast, the dimensions for organizational influence in this paper emerge from the bottom up, directly from the components of the principled decision-theoretic framework.

Much of the literature in multiagent organization design and specification concentrates on formulating organizational modeling languages (OMLs), such as MOISE$^+$ [13] and OMNI [14], among a variety of others. Though the specifics of these OMLs vary, they generally emphasize specifying an agent organization at an abstract level in terms of roles, role relationships/interactions, norms, etc. They also tend to be agnostic about how an agent would map the abstract specification into its internal reasoning processes. Hence, our work here complements that work, helping to bridge the gap between modeling and implementation by identifying opportunities and limitations in what OMLs can express that can be meaningfully mapped into influences over decision-theoretic agents.

4. BASELINE ORGANIZATION

In our preceding characterization of how an organizational designer influences an agent, the basic idea is that the design $\theta_i = \langle S_{\theta_i}, \alpha_{\theta_i}, A_{\theta_i}, R_{\theta_i}, P_{\theta_i}, T_{\theta_i} \rangle$ supplants the agent's "local" model $\mathcal{M}_i = \langle S_i, \alpha_i, A_i, R_i, P_i, T_i \rangle$. But where does an agent's (original) local model come from? Clearly, the performance improvements that an organizational design will make depends on how (dis)organized the agents are when following their initial local models. This means that we could show arbitrarily good performance improvements by initializing agents with arbitrarily bad local models.

This is a fundamental and under-addressed quandary in the artificial agent organizations research field. The combination of initial local models of agents essentially *do* comprise an organizational design. When assembling an agent system, agents might be selected based on the inherent alignment between their local models and the (organizational) biases of whomever is assembling the system. The actions agents are capable of, the states they can represent, their predispositions about what states are rewarding, etc. can all factor into decisions about which agents are included in the system.

Our evaluation of the improvement achievable by following a designed organization thus depends on defining a baseline organization. To develop as even-handed a baseline as possible, we advocate initializing local decision models by performing an uninformed mapping of the joint Dec-POMDP models into localized versions. In this way, the local models are perforce aligned with the global model, but they are not crafted to differentiate the roles and behaviors of the agents. Essentially, the philosophy is to endow each agent with a local model that directly makes the individual agent responsible for solving the global problem, to the extent its awareness and capabilities allow.

Specifically, our methodology for initializing agents' local models to provide an experimental baseline is as follows. First, we assume that the subset of state features directly observable to the agent defines its local state representation. Second, the action space of an agent is simply its component of the joint action space. Third, the local reward function is the same as the global reward function, except that any components involving features outside of the agent's local state representation are dropped, since the agent does not have values for those features. Fourth, the local transition model corresponds to the joint transition entries where the existence of other agents is moot. Finally, the initial local state distribution maps the global distribution into the local state space, and the local finite time horizon is identical to the global value. In the firefighting domain, the baseline organization is the local model as we described it in the last paragraph of Section 2.

While this method for creating a baseline model is still dependent on somewhat arbitrary decisions (e.g., which features are included in an agent's local state), the idea is that aspects that influence how an agent formulates a policy (what is rewarding, what might happen in the world, etc.) are aligned with the "true" global model but contain as little information as possible about what an agent might expect others to do in the world. We assume that it is up to an organizational designer to provide such information.

Despite our adoption of this uninformed-but-aligned baseline, we have recognized that other factors also influence the difference that organizational design can make. A simple example we've encountered is how the initial configuration of state can greatly affect whether the baseline organization is effective. In the firefighting domain, if we assume that the fires pop up across the space with uniform probability, then where should we assume firefighters begin? If we assume that they are uniformly distributed in the environment, then their local models (where they prefer fighting nearby fires) inherently lead to a good allocation of tasks (fires) to agents. If we assume that they all start in the same location, on the other hand, then the local models inherently lead to agents moving around *en masse* and yields no parallelism benefits.[2] Even randomly placing firefighters is not an answer, because distributing fires and agents in the same uniformly random way introduces its own bias. In our experiments described in Section 5.2, we present results from the two extreme environments: the agents beginning uniformly distributed; and the agents beginning clustered in the center of the grid world, which represents the best and worst case in expectation for the baseline organization respectively.

5. EVALUATION

We now turn to evaluating our claimed benefits of characterizing the organizational design space in terms of adjusting the components of an agent's decision-theoretic model. In this section, we use our simplified firefighting problem domain to investigate the effects of modifying each component individually, and in combination, as a step toward building an automated design algorithm.

[2]Note that if multiple firefighters on the same fire had a super-additive effect, instead of the sub-additive effect in our domain, then initially spreading out could be disadvantageous, while moving around in a pack might be beneficial.

The experiments that follow use the problem formulation already described (Section 2), in terms of state features, agents' actions, their transitions, and joint reward function. To test the degree to which an organizational design provides long-term benefit to a multiagent system, we run a fixed organizational design over a large number of randomly-generated problem instances, where each instance is an episode that begins with a randomized configuration of fires and ends when the time horizon is reached. By the luck of the draw, some problem instances might be well suited to one organization over another. We focus on aggregate performance over many episodes not only to smooth out the randomness of the instances but moreover to identify an organization's effectiveness over the long term, due to the assumption that organizational design has a high cost that is amortized over time. The measures of performance of interest are the expected joint reward and the planning overhead of the agents in each episode. A well-designed organization is one that improves joint reward while also simplifying each agent's local planning problem.

5.1 Comparison to Optimal

To be able to compute an upper-bound on performance (an optimal joint policy) against which to compare, we begin with problems in a simple 10×5 grid world with 2 cooperative agents and 2 fires, as illustrated in Figure 1. The distribution of fires' locations is uniformly random over the entire grid, and the fires' intensities are uniformly random over $\{1, 2, 3\}$; however, the agents always begin in the same locations (those in Figure 1). To speed up the tests without pruning any viable solutions, the finite time horizon is the maximal time either agent would require to put out both fires alone (varies per episode). To get a sense of the impact of different organizational designs, we tested three designs in addition to the baseline organization. One, called fullOverlapOrg, assigns both agents to be responsible for all fires in the entire grid. However, unlike the baseline organization where agents have no model of each other, fullOverlapOrg provides agents with improved transition models that reflect the possible activities of the other agent. Specifically, our organizational designer heuristically assumes that an agent will first fight the fire closest in its region, then the closest fire from there, and so on, until the time horizon. So, the organization adjusts the other agent's transition function to anticipate that some fires (on the other side of the grid) will have decreasing intensities even without fighting them itself, helping it refrain from rushing to distant high-intensity fires that will be addressed by someone else.

A second organizational design, called partitionOrg, partitions the locations, assigning responsibility for fires in the western 5×5 subgrid to A1, and the eastern subgrid to A2, removing actions from the agents' action spaces that would move them out of their regions. More generally, partitionOrg represents an assignment of each task to exactly one agent.

The third organization is called smallOverlapOrg, in which the 4 middle columns of the grid are in both agents' regions of responsibility. Like in partitionOrg, agents' action spaces are pruned so an agent doesn't consider moving out of its region, while like fullOverlapOrg, an agent has an adjusted transition function to reflect that fires in its local state space have a chance of going out without it fighting them.

To create the local policies, each agent uses its organizational model to create the reachable state space from the given initial state forward. It then uses CPLEX [7] to calculate the optimal local policy for the reachable state space using the linear program as formulated by Kallenberg [9].

Before describing our results, we have to address one more issue. Agents build policies that only consider states they could conceivably reach within the time horizon. Because an agent using the baseline organization models the world as if it is alone, its reachable state space does not include states where some fires' intensities decrease without it fighting them. Thus, when executing its policy it could reach an unexpected state. Rather than explode the state space by including low-probability transitions covering every possibility, in our experiments we simply assume that when an agent "falls off" its policy (reaches an unplanned state), it constructs a new policy going forward from its (unexpected) current state, and that this planning is instantaneous with respect to events in the world. (The world "waits" for the agent to replan.) While future work should treat this more realistically, for the purposes of our experiments this assumption favors less informed organizations (that fall off policy more frequently) more than informed ones, so the benefits of organizational design will be, if anything, understated. Finally, note that agents given improved transitions might still sometimes fall off policy, because the heuristics used in the transition functions are imperfect.

Our experiments are summarized in Table 1. We generated 1,500 episodes with random initial states and solved each using the 3 organizational designs (partitionOrg, smallOverlapOrg, and fullOverlapOrg), as well as the uninformed baseline organization. We also generated the optimal joint policy for each episode to compute the optimal attainable reward if the agents could afford the time to generate it and could also sense each others' positions. These results show that even simple organizational designs can improve rewards considerably compared to the baseline, but that overly restrictive organizations (partitionOrg) can degrade performance because the same agent too often must fight both fires. As one would expect, more restrictive organizations increasingly simplify agents' local decision problems. Moreover, note that all of the organizations decrease local computation over the baseline, because in the baseline both agents solve larger problems (putting out all the fires by themselves) than when they are informed (through the transition function) that they will have help.

Note that the fullOverlapOrg has greater global awareness than the other organizations; however, this increased awareness incurs greater computational costs. Because performance is basically inversely correlated with computation, we created a unified performance metric by adopting the standard methodology of having the agents sit idle at the start of execution while they create their policies. To do this, we convert the actual CPU time for policy creation into simulation time steps, and then force the agents to sit idle for that many time steps at the beginning of the episode (essentially performing NOOPs). Figure 2 presents the adjusted expected reward after accounting for computational costs as a function of the CPU time per simulation time step. These results confirm our intuitions that when computation is expensive (low c) paritionOrg is best due to its highly simplified decision process. Then as computation becomes cheaper (c increases), the more flexible organizations become superior, and finally when computation is very cheap, computing the optimal joint policy becomes best.

	Reward	Plan Time	Replans
Baseline	-15.97	86	1.32
ParitionOrg	-16.15	12	0.26
SmallOverlapOrg	-14.74	27	0.16
FullOverlapOrg	-14.70	70	0.14
Joint	-14.37	24558	0.00

Table 1: Mean experimental results for Section 5.1 for expected reward, CPU time to create initial policy (ms), and average number of times the replanning mechanism was invoked per agent per episode.

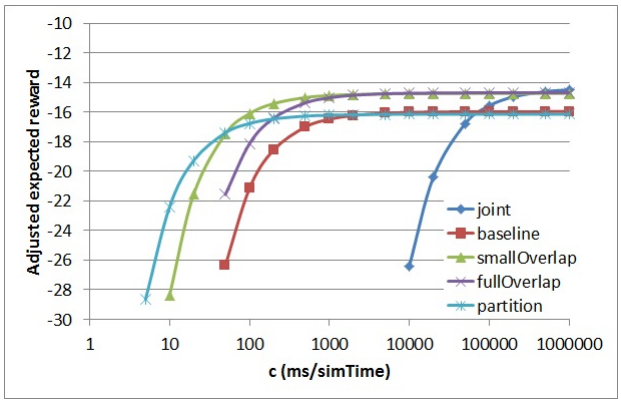


Figure 2: Adjusted rewards for Section 5.1 after accounting for computation time as a function of the CPU time per simulation time step.

5.2 Design Components

We now turn to isolating the impact of different dimensions of organizational design, corresponding to different components of the agents' decision models. For these experiments, we use larger environments with 10 cooperative agents and 10 fires on a 25×10 grid. Fires are still distributed uniformly randomly over the entire grid, with intensities drawn uniformly from $\{1, 2, 3\}$. As discussed at the end of Section 4, the initial locations of the agents can favor, or disfavor, some organizational designs. Thus, in these experiments, we consider two extreme cases of initial locations for the agents: where they are evenly spread around the environment; and where they are clustered at the center of the grid.

To understand the impact of designing along different dimensions, we implemented largely the same organizational structure using the different components. The structure inherits from the smallOverlapOrg in Section 5.1, narrowing agents' ranges of policies to consider while still providing them with some flexibility to load balance by having overlapping regions of responsibility. Specifically, the 25×10 grid is divided into 10 distinct 5×5 subgrids, one for each agent, to act as the agent's primary area of responsibility (PAR). In each (non-wall) direction, the subgrid is expanded by 3 cells to introduce overlap; conceptually, this is an agent's secondary area of responsibility (SAR). We implemented 5 organizations capturing this fundamental structure: **actionOrg** removes actions that take an agent out of its combined PAR and SAR; **stateOrg** removes features for states outside of the combined PAR and SAR; **rewardOrg** penalizes the agent with increasing severity for leaving its PAR (Manhattan distance from PAR squared); **transitionOrg** models how fires in the PAR and SAR might go out due to someone else's actions using the same heuristics as in Section 5.1 (and like stateOrg ignores more distant fires to curb state-space explosion resulting from the richer transition model); and **fullOrg** uses all of the dimensional levers just described.

We generated 100 random episodes (initial fire configurations), for each of the spread and clustered variations of agents' initial locations. For each episode, we ran each of the 5 organizations above, as well as the baseline organization. The problems were too large to compute optimal joint policies. Table 2 presents the results for these experiments.

These results illustrate many of the intuitions from Section 3.1. As others have discovered, reward shaping can be a powerful tool for increasing the expected joint reward; however, it does not generally reduce the agents' computational efforts. Shaping the transition functions can also yield a large increase in the expected reward; however, it substantially increases the agents' computational costs. Notice that organizations with improved transition functions replan during execution much less, indicating that if recovering from falling off policy incurs non-negligible cost, then transition shaping could be of critical importance. We also observe that constraining the agents' action or state spaces can greatly simplify the agents' decision problems and can also increase the expected joint reward. Finally, with fullOrg, we observe that the organizational influences in the components are not completely redundant, as it is largely possible to obtain the additive benefits found in each of the other organizations. The drop in expected reward as compared to transitionOrg is due to the shaped reward functions urging agents to quickly go their respective PARs rather than stop and fight fires along the way. However, also note that the computation time is drastically reduced, suggesting that the tradeoff would be beneficial unless computation is exceptionally cheap.

Finally, the reader may have noted that our experiments did not evaluate the impact of restructuring the other two components: the initial state distribution α_{θ_i}; and the time horizon T_{θ_i}. One could envision organizations that modify T_i to give agents specific roles for planning horizons, where some agents focus on the near-term and others on the long-term, though the organization would probably also focus an agent's action space A_{θ_i} on actions of a matched granularity. If α_i summarizes the exogenously-determined initial state, the designer can only map this into the agent's adjusted state space S_{θ_i}, as was implicitly done for the organizational variations above. However, as seen in the relative performance between the spread and clustered environments, if the organization can impose initial states on agents (spreading them out in anticipation of arising fire configurations), then this provides an additional lever for influencing collective performance.

6. ORGANIZATIONAL ADEPTNESS

As demonstrated in Section 5.1, an organizational designer confronts tradeoffs in deciding how tightly to influence the agents. If not tightly enough, agents might duplicate effort or work at cross purposes while, if too tightly, agents might load balance poorly or have tasks fall between the cracks. We assume the organizational designer can use a model of the expected problem distribution to form an organization that, in expectation, will work best. However, if its model is (or over time becomes) erroneous, the agents must decide how to refine, revise, or even abandon that organizational structure.

	Large Problems (Spread)			Large Problems (Clustered)		
	Reward	**Plan Time**	**Replans**	**Reward**	**Plan Time**	**Replans**
Baseline	-107.40	1646	7.89	-436.7	10912	0.00
RewardOrg	-91.45	1817	7.49	-242.0	11051	9.38
TransitionOrg	-85.14	14606	0.86	-222.5	10859	0.55
ActionOrg	-94.14	551	7.70	-264.5	621	8.56
StateOrg	-94.14	1237	2.60	-254.4	1588	1.50
FullOrg	-87.51	5476	0.88	-250.4	2652	1.02

Table 2: Mean experimental results for Section 5.2 for expected reward, CPU time to create initial policy (ms), and average number of times the replanning mechanism was invoked per agent per episode.

Following Corkill *et al.* [3], we refer to agents with this capability as *organizationally adept agents* (OAAs). As advocated elsewhere [5], agents need *operational control* capabilities to elaborate and refine organizational control guidelines. For example, agents with overlapping areas of responsibility can use operational control to resolve who is responsible for which tasks in the current situation. But operational control can be expensive (in computation, communication, delay, etc.), so organizations that more narrowly define the roles of each agent, and thus require less operational control, can be preferable. However, if the designer's assumptions about the problems that will be encountered are wrong, a narrower organization might leave too little latitude for operational refinement to meet coordination needs. An OAA should be able to compare the problems actually encountered to the organizational designer's expectations, and decide whether a change or abandonment of organization is warranted, thus allowing for narrower organizations to be utilized.

Our decision-theoretic formulation of organizational design provides a framework for agents to make such comparisons and decisions, and thus for a more formal characterization of what it means for an agent to be organizationally adept. For example, an OAA i can compare its organizational initial state distribution α_{θ_i} with the initial states it has actually witnessed over a series of episodes to detect mismatches. Similarly, i can recognize that, for example, the probabilities that fires will be be put out by others according to P_{θ_i} are not supported by statistics over observed transitions, or that states whose rewards have been shaped by the organization are seldom reachable.

When the expectations implied in the organizational structure stem from high-level assumptions the designer has about the problem domain, such as that fires will appear uniformly randomly through the entire region, the designer can annotate an organization with the assumptions on which its selection is conditioned. Our current decision-theoretic OAA architecture captures such annotations in terms of variables to monitor and expectations over their values. More formally, optionally along with its organizational specification θ_i, agent i can receive a set of monitor-variable and value-expectation pairs $\psi_i = \langle(\psi_{i_1}, v_{i_1})...(\psi_{i_m}, v_{i_m})\rangle$. (If none are provided, the OAA can still use the expectation implicit in θ_i.) Essentially, the annotated formulation indicates that, to the extent that the monitor-variables take on values consistent with expectations, the organization should be followed.

As a preliminary illustration of these OAA concepts, we use our 10-agent problem domain from Section 5.2 and consider two different models of how fires arise: having an increasingly higher probability of arising toward the east end of the grid; and having an increasingly higher probability of arising toward the west end. Note that the desired organizational behavior is significantly different between the two environments; in the eastEnvironment we would want to designate more agents to the eastern region (and *vice versa* for the westEnvironment). We designed a specialized organization for each case, which are analogous to fullOrg from Section 5.2 except that the PARs are non-uniformly sized to compensate for the biased fire distributions. For example, in the westOrg, 3 agents are responsible for the western 4 columns (4×3, 4×4, and 4×3 PARs). Working eastward, the PARs get progressively larger, starting with two 4×5 PARs (stacked vertically), then two 5×5 PARs, then two 6×5 PARs. Finally, a lone agent is responsible for the eastern edge with a 5×10 PAR. The eastOrg is a symmetric copy of the westOrg. Associated with each organization is a set of monitor variables informing each agent that the organizational designer expected one fire, on average, to be in its PAR (for that organization).

We provided the agents with both of these annotated organizations, in addition to fullOrg, which is weakly applicable both environments. The agents all initially adopt (based on the designer's directives) fullOrg to reflect the designer's uncertainty about the environment. As episodes are experienced, the agents track their monitor variables. They then jointly aggregate this observational evidence, e, and perform Bayesian inference to calculate the likelihood that each of the environments is the actual environment being observed, which are used to estimate the expected reward of following each available organization. The agents then collectively and greedily adopt the organization with the highest anticipated expected reward. Formally, they adopt Θ^*:

$$\Theta^* = \arg\max_{\Theta} E[R|\Theta, e] - c(\Theta_c, \Theta)$$

$$E[R|\Theta, e] = \sum_j Pr(M_j|e) E[R|\Theta, M_j]$$

where $c(\Theta_c, \Theta)$ is the cost of switching from the current organization Θ_c to Θ. We assume there is no cost for remaining in the same organization, $\forall i\ c(\Theta_i, \Theta_i) = 0$. $Pr(M_j|e)$ is the likelihood of environmental model M_j being the actual model given e, which the agents calculate via Bayesian inference. $E[R|\Theta, M_j]$ is the expected reward of following organization Θ in M_j, which we assume is provided by the organizational designer in the annotations. For our experiments, we estimated $E[R|\Theta, M_j]$ by *a priori* simulating Θ on a training set of episodes created from M_j.

Our experiments present the agents with episode batches where the true environment model is selected uniformly randomly from the two environments every 20 episodes (all organizations face the same episodes in the same order).

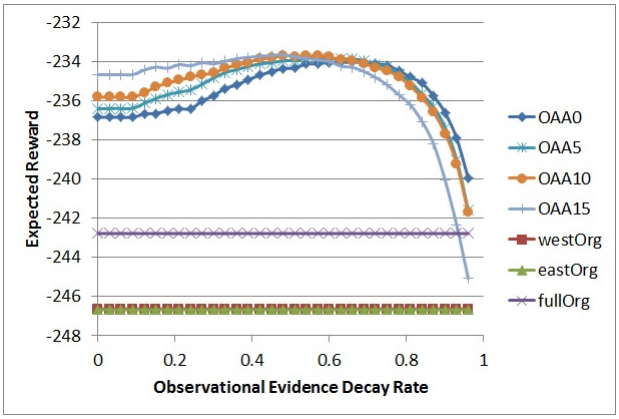


Figure 3: Expected reward as a function of the observational evidence decay rate.

Only at the end of each episode are the agents allowed to collectively adopt whichever organization they deem best. Since the true environment is dynamic, we allow the organizational designer to set a decay rate in the annotations, which the agents use to decay the importance of past monitor variable observations. We performed experiments with several organizations: statically using the east/west/fullOrg for every episode; and several parameter settings of the OAA process described above. OAAx refers to the OAA process above where the organizational switching cost is x.

Our results are summarized in Figure 3, which confirms several intuitions. Firstly, statically following either specialized organization performs poorly since they suffer when being used in the environment they were not intended for; however, statically following fullOrg makes a noticeable improvement by being weakly suited to both environments. Secondly, by allowing the agents to react to the shifting environment, the OAA capability (in general) can yield a large performance gain. Finally, if the organizational switching cost is low, the agents should maintain sufficient observational evidence history in order to prevent the agents from switching organizations due to a transient episode, such as when an episode from the eastEnvironment happens to "look" like an episode from the westEnvironment due to unlikely fire locations.

7. CONCLUSIONS

In this paper we have presented a decision-theoretic framework that provides a systematic method for enumerating the possible ways in which an organization can influence agents' decision-making processes. We have intuitively described and empirically demonstrated how influencing the various Dec-POMDP components can both increase the agents' expected joint reward as well as simplify their local decision problems as compared to a baseline local model. Finally, in Section 6 we have shown how our organizational framework provides a more formal characterization of what organizational adeptness can mean compared to prior work and have provided preliminary empirical evidence of the benefits of OAA. In the future, we plan to expand the functionality of OAA; for example, rather than greedily reacting to current model likelihoods, the agents could make predictions about the ways the environment is changing and preemptively switch organizations. Additionally, we plan to investigate the effects of an agent reasoning unilaterally about its observational evidence and individually changing its organization (as opposed to a central decision process), as well as the possibility of gradually blending organizations together when switching as opposed to the all-or-nothing switching described in this paper. Finally, using the insights gained from Section 5, we plan to develop an automated organizational designer that can create organizations within our structured framework.

8. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their thoughtful comments, and our collaborators at the University of Massachusetts for their consistently helpful feedback. This work was supported by NSF grant IIS-0964512.

9. REFERENCES

[1] A. K. Agogino and K. Tumer. Multi-agent reward analysis for learning in noisy domains. In *AAMAS*, pages 81–88, 2005.

[2] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research*, 27(4):819–840, 2002.

[3] D. Corkill, E. Durfee, V. Lesser, H. Zafar, and C. Zhang. Organizationally Adept Agents. In *COINS2011 Workshop at AAMAS*, 2011.

[4] D. D. Corkill and S. E. Lander. Diversity in Agent Organizations. *Object Magazine*, 8(4):41–47, 1998.

[5] E. H. Durfee and Y. p. So. The effects of runtime coordination strategies within static organizations. In *IJCAI*, pages 612–619, 1997.

[6] M. S. Fox, M. Barbuceanu, M. Gruninger, and J. Lin. An organizational ontology for enterprise modeling. In *Simulating organizations*, pages 131–152. MIT Press, Cambridge, MA, USA, 1998.

[7] IBM. IBM ILOG CPLEX, 2011. See http://www-01.ibm.com/software/integration/optimization/cplex-optimizer/.

[8] N. R. Jennings. Commitments and conventions: The foundation of coordination in multi-agent systems. *The Knowledge Engineering Review*, 8(03):223–250, 1993.

[9] L. C. M. Kallenberg. *Linear Programming and Finite Markovian Control*. Mathematical Centre Tracts, 1983.

[10] A. Y. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, pages 278–287, 1999.

[11] Y. Shoham and M. Tennenholtz. On social laws for artificial agent societies: off-line design. *Artificial Intelligence*, 73(1-2):231 – 252, 1995.

[12] Y. p. So and E. H. Durfee. Designing organizations for computational agents. In *Simulating Organizations*, pages 47–64. MIT Press, Cambridge, MA, USA, 1998.

[13] M. B. van Riemsdijk, K. V. Hindriks, C. M. Jonker, and M. Sierhuis. Formalizing organizational constraints: A semantic approach. In *AMMAS*, pages 823–830, 2010.

[14] J. Vázquez-Salceda, V. Dignum, and F. Dignum. Organizing multiagent systems. *Autonomous Agents and Multi-Agent Systems*, 11:307–360, November 2005.

[15] D. Wolpert and K. Tumer. Optimal payoff functions for members of collectives. *Advances in Complex Systems*, 4(2/3):265–279, 2001.

Reasoning under Compliance Assumptions in Normative Multiagent Systems

Max Knobbout
Utrecht University
Dept. of Computer Science
The Netherlands
M.Knobbout@students.uu.nl

Mehdi Dastani
Utrecht University
Dept. of Computer Science
The Netherlands
M.M.Dastani@uu.nl

ABSTRACT

The use of norms in multiagent systems has proven to be a successful approach in order to coordinate and regulate the behaviour of participating agents. In such normative systems it is generally assumed that agents can obey or disobey norms. In this paper, we develop a logical framework for normative systems that allows reasoning about agents' abilities under a multitude of norm compliance assumptions. In particular, we investigate different types of norm compliance and propose an extension of Alternating Temporal Logic (ATL) to reason about the abilities of (coalitions of) agents under different types of norm compliance assumptions. For this extension we show that the problem of model-checking remains close to the domain of standard ATL. Finally, we show that some norms can limit an agent's autonomy in the sense that an agent cannot control the violation of these norms. We present and discuss various classes of the so-called self-supporting norms, i.e., norms for which individual agents have control over their violations.

Categories and Subject Descriptors

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

General Terms

Theory, Design, Verification

Keywords

Normative Systems, Organizations, Verification, Logic

1. INTRODUCTION

The use of norms and social laws in multiagent systems has proven to be a successful approach in order to ensure the overall objectives of such systems. Early examples of such an approach can be found in Shoham and Tennenholtz [7] and Moses and Tennenholtz [6]. In these works, social laws are used to constrain the behaviour of the agents by forbidding certain actions in specific situations. This line of research was later extended by several authors in a multitude of ways. For example, in [8] the basic idea is extended to the more expressive modelling domain of alternating temporal logic (ATL) with the idea that the application of a social law in a multiagent system is successful if, by implementing it, the overall objective of the system is satisfied. Later these ideas were extended with the notion of preference to reason about norm compliance in normative systems. Example of such extensions are [9], where each agent was attributed a single goal, and [2], where agents were given a priority list of goals. The main topic of these papers was to investigate whether certain sets of norms can be considered 'stable' under consideration of the agents' preferences. Another extension of these ideas was introduced in [4], where norms are considered from a mechanism design perspective. In this framework, norms are related to game theoretic solution concepts such that one can specify and verify whether a set of norms in a multiagent system implements some social/overall objectives in specific equilibria.

In this paper, we introduce an extension of ATL to reason about properties of normative multiagent systems under various norm compliance assumptions. In our setting, norms are assumed to be directed to coalitions of agents, e.g., a norm states that a set of agents should not take certain actions in specific states. Moreover, we assume that agents can disobey norms in the sense that they can take actions that are disallowed by the norms. We then define various types of norm compliance behaviors such as "a coalition of agents obey/disobey norms that are directed to precisely this coalition" or "a coalition of agents obey/disobey norms that are directed to this coalition and all its subcoalitions". Our proposed extension of ATL can be used to specify and verify whether some overall objective of multi-agent systems can be satisfied under the assumption that a coalition of agents behave according to a specific norm compliance type while the agents outside the coalition behave according to another norm compliance type. The proposed extension can be used by the designers of normative systems to analyze norms that are directed to coalitions of agents and to investigate their impacts on the overall system behaviour under the assumption of specific norm compliance types. This extension can also be used by individual agents who need to reason and decide if they can achieve their own objectives by behaving according to a specific norm compliance type in a normative system when the system is populated by other agents that behave according to another specific norm compliance type.

In our framework, we introduce abstract normative constraints as a basic extension of the social law paradigm. We consider norms as similar to social laws in the sense that

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

they denote the disallowed actions of the agents. However, norms are considered as different from social laws in the sense that the agents are allowed to violate them. Moreover, and in contrast to social laws, norms are directed to coalitions of agents and not only to individual agents. Thus, we do not assume that "implementing" a system of norms enforces every agent to be perfectly norm obedient. To some extent, our research is related to [1] in which the need for robust normative systems is discussed. They introduce an extension of CTL (Computation Tree Logic) in which statements such as "if coalition C is norm compliant, then this is sufficient to guarantee φ" can be expressed. In their work a robust normative system is defined as a multiagent system which remains 'effective' (specified by some criterium) even if certain agents behave in a non-compliant manner. In our framework, a more expressive language is proposed to also reason about system properties under various norm compliance assumptions. An important difference with this paper is that we depart from the domain of 'agent-labelled Kripke Structures' to the more expressive and natural domain of Concurrent Game Structures. Using our proposed logic, we can then construct such statements as "does there exist a norm compliant strategy for a given agent to guarantee φ under the assumption that the other agents are non-compliant?". We believe that this extra layer of expressivity is indeed important for agents in order to decide whether to comply with the given norms or participate in the multiagent system. Our framework can also be useful from the perspective of a designer when trying to design systems which formally abide certain properties under certain norm compliance assumptions.

In section 2 we will give a quick recap on the syntax and semantics of standard ATL. In section 3 we introduce the notion of *abstract normative constraints* and in section 4 we provide the syntax and semantics of our proposed ATL extension with an elaborate example. In section 5 we discuss the model checking complexity of the proposed ATL extension. Finally, in section 6 we present the notion of self-supporting norm sets.

2. CONCURRENT GAME STRUCTURES

In this section, we first briefly recall on the definition of Concurrent Game Structures, the underlying model we use for multiagent systems. A Concurrent Game Structure, or CGS, can be seen as a multiagent extension of a simple transition system. It consists of states of the world, and a complete labelling of joint-actions over the transitions connecting these states. More formally, a CGS is a tuple $S = \langle k, Q, \Pi, \pi, Ac, \delta \rangle$ such that:

- A natural numbers $k \geq 1$ of players. In our model, each player corresponds to a number. We sometimes use Σ to talk about the set $\{1, ..., k\}$.
- A finite set Q of states.
- A finite set Π of atomic propositions.
- A mapping π which maps each state $q \in Q$ to a subset of propositions which are true at q. Thus for each $q \in Q$ we have $\pi(q) \subseteq \Pi$.
- A mapping Ac which maps each player $a \in \Sigma$ and each state $q \in Q$ to a non-empty subset of the natural numbers denoting the moves for player a in state q. Thus for each $a \in \Sigma$ and each $q \in Q$ it holds that $Ac(a, q) \in \mathcal{P}(\mathbb{N})$. In our model, each action corresponds to a natural number. For each state $q \in Q$, a move vector is a tuple $\langle \alpha_1, ..., \alpha_k \rangle$ such that $\alpha_i \in Ac(i, q)$. The set of all move vectors for a state $q \in Q$, denoted by $D(q)$, is given by $Ac(1, q) \times ... \times Ac(k, q)$.
- A mapping δ which maps each state $q \in Q$ and each move vector $\langle \alpha_1, ..., \alpha_k \rangle \in D(q)$ to another state that results from state q if each player adopted the move denoted in the move vector. Thus for each $q \in Q$ and each $\langle \alpha_1, ..., \alpha_k \rangle \in D(q)$ we have $\delta(q, \langle \alpha_1, ..., \alpha_k \rangle) \in Q$.

Note that this model is synchronous, meaning that at any moment in time each agent needs to decide on an action synchronously. Moreover, it is also deterministic; the same action in the same state will always yield the same resulting state.

Alternating-time temporal logic, as discussed in [3], is interpreted over a concurrent game structure S that has the same propositions and players. Evaluating a propositional formula at a given state amounts to verifying whether the formula is satisfied given the labelling of that state. To evaluate a formula of the form $\langle\langle A \rangle\rangle \psi$ at a state q of S, we can consider a game between a protagonist and an antagonist which results in a computation. At every round the protagonist chooses for each player in A a move, and then the antagonist proceeds by choosing for every player $\Sigma \backslash A$ a move, after which the position is updated from q to q'. This process is repeated indefinitely, which results in a computation λ. The protagonist wins the game if the resulting computation satisfies the subformula ψ, which is a temporal formula of the form $\bigcirc\varphi$, $\Box\varphi$ or $\varphi_1 \mathcal{U} \varphi_2$ (where φ,φ_1,φ_2 are again ATL formula's), otherwise the antagonist wins. Then the formula $\langle\langle A \rangle\rangle \psi$ is satisfied at q if the protagonist has a winning strategy for this game.

More formally, ATL is characterized by the following grammar, where $p \in \Pi$ and $A \subseteq \Sigma$: $\varphi ::= p | \neg\varphi | \varphi \wedge \varphi | \langle\langle A \rangle\rangle \bigcirc \varphi | \langle\langle A \rangle\rangle \Box \varphi | \langle\langle A \rangle\rangle \varphi \mathcal{U} \varphi$. In order to define the semantics, we first have to define the notion of strategy. A strategy for a player $a \in \Sigma$ is a mapping s_a which maps a finite (non-empty) sequence of states to an action belonging to the last state of this sequence. Thus for each sequence $q_0, ..., q_k \in Q^+$ we have $s_a(q_0, ..., q_k) \in Ac(a, q_k)$. Given a set of players $A \subseteq \Sigma$ and a state $q \in Q$, let $S_A = \{s_a \mid a \in A\}$ be the set of strategies A adopt and let $out(q, S_A)$ be the set of computations starting from state q which the players in A can enforce by following their respective strategies (that is, independent of what the players $\Sigma \backslash A$ play). A computation $\lambda = q_0, q_1, q_2, ...$ is in $out(q_0, S_A)$ if it holds that for all positions $i \geq 0$ there is a move vector $\langle \alpha_1, ..., \alpha_k \rangle \in D(\lambda[i])$ (where $\lambda[i]$ denotes the state at position i) such that $\delta(\lambda[i], \langle \alpha_1, ..., \alpha_k \rangle) = \lambda[i+1]$ and for all $a \in A$ it is the case that $s_a(\lambda[0, i]) = \alpha_a$.

Given a game structure $S = \langle k, Q, \Pi, \pi, Ac, \delta \rangle$ and a state $q \in Q$, we define the semantics inductively as follows:

- $S, q \vDash p$ for any proposition $p \in \Pi$ iff $p \in \pi(q)$.
- $S, q \vDash \neg\varphi$ iff $S, q \nvDash \varphi$.
- $S, q \vDash \varphi_1 \wedge \varphi_2$ iff $S, q \vDash \varphi_1$ and $S, q \vDash \varphi_2$.
- $S, q \vDash \langle\langle A \rangle\rangle \bigcirc \varphi$ iff there exists a strategy set S_A for A such that for every computation $\lambda \in out(q, S_A)$ it holds that $S, \lambda[1] \vDash \varphi$.

- $S, q \vDash \langle\!\langle A \rangle\!\rangle \Box \varphi$ iff there exists a strategy set S_A for A such that for every computation $\lambda \in out(q, S_A)$ and all positions $i \geq 0$ it holds that $S, \lambda[i] \vDash \varphi$.

- $S, q \vDash \langle\!\langle A \rangle\!\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there exists a strategy set S_A for A such that for every computation $\lambda \in out(q, S_A)$ there exists a position $i \geq 0$ such that for all positions $0 \leq j < i$ it holds that $S, \lambda[j] \vDash \varphi_1$ and $S, \lambda[i] \vDash \varphi_2$.

A formula of the form $\langle\!\langle A \rangle\!\rangle \psi$ should intuitively be read as "Coalition A has a strategy in order to enforce ψ", where ψ can be a temporal formula of the form $\bigcirc\varphi$, to be read as "in the next state φ", $\Box\varphi$, to be read as "always in the future φ" and $\varphi_1 \mathcal{U} \varphi_2$, to be read as "$\varphi_1$ untill φ_2 starts to hold".

3. ABSTRACT NORMATIVE CONSTRAINTS

Before we discuss the notions of compliance and non-compliance in normative systems, it is important to clarify what we understand under a normative multiagent system. For our purposes, much in line with previous research seen in e.g. [2], a normative system is simply a set of constraints on the behaviour of the agents. However, since we have entered the domain of concurrent game structures, we extend this notion to not only account for behaviour of agents, but also behaviour of coalitions. Moreover, we also allow the agents to violate these constraints; they are not hard-constraints on the behaviour of the agents. More precisely, given a game structure $S = \langle k, Q, \Pi, \pi, Ac, \delta \rangle$, an abstract normative constraint $\langle A, \gamma \rangle$ is a tuple consisting of a subset of player $A \subseteq \Sigma$ and a mapping γ which maps a player $a \in A$ and state $q \in Q$ to a set of actions for each player that can be taken in that state. Thus given a state $q \in Q$ and a set of player $A \subseteq \Sigma$, for all $a \in A$ we have that $\gamma(a, q) \subseteq Ac(a, q)$. The set $\gamma(a, q)$ denotes all the actions that are normatively demotivated in state q for agent a. Given a set of abstract normative constraints Γ, we define $\Gamma_A = \{\langle X, \gamma \rangle \in \Gamma \mid X = A\}$ and $\Gamma_A^- = \{\langle X, \gamma \rangle \in \Gamma \mid X \subseteq A\}$. In words, Γ_A is the set of abstract normative constraints only applicable to exactly A and Γ_A^- the set of abstract normative constraints only applicable to A or any sub-coalition of A. Given a computation $\lambda = q_0, q_1, q_2, ...$ and a set of abstract normative constraints Γ, we say that $\langle A, \gamma \rangle \in \Gamma$ is enabled for A at position $i+1 \geq 0$ if $\forall a \in A$ it holds that $\gamma(a, \lambda[i]) \neq \varnothing$ and taken at position $i+1 \geq 0$ if there is a move vector $\langle \alpha_1, ..., \alpha_k \rangle \in D(\lambda[i])$ such that $\delta(\lambda[i], \langle \alpha_1, ..., \alpha_k \rangle) = \lambda[i+1]$ and $\forall a \in A$ it holds that $\alpha_a \in \gamma(a, \lambda[i])$. Notice that by this interpretation, an abstract normative constraint can still be taken even though the actual action an agent performed along a computation differs from the one prescribed by γ.

Given a set of agents A, we can give different interpretations towards obedience with respect to Γ. Below we define 7 different norm compliance types: 3 types of obediences, 3 types of disobediences, and 1 type of neglectfulness. However, we stress that this primarily is a choice; certainly more types of obediences can be considered and constructed based on the logical framework we provide.

1. **Coalitional obedience** A computation λ is *coalitional obedient* with respect to a set of agents A and an abstract normative constraint set Γ, or $\langle \Gamma, A, c\text{-}ob \rangle$-*obedient*, if at any point in the computation for every $\langle A', \gamma \rangle \in \Gamma_A^-$ it is the case that if $\langle A', \gamma \rangle$ is enabled in λ, then $\langle A', \gamma \rangle$ is not taken.
2. **Total/Selective individual obedience** A computation λ is *total individual obedient* with respect to a set of agents A and an abstract normative constraint set Γ, or $\langle \Gamma, A, t\text{-}ob \rangle$-*obedient*, if at any point in the computation for all $a \in A$ it holds that for every $\langle \{a\}, \gamma \rangle \in \Gamma_{\{a\}}$ it is the case that if $\langle \{a\}, \gamma \rangle$ is enabled, then $\langle \{a\}, \gamma \rangle$ is not taken. A computation λ is *selective individually obedient* with respect to a set of agents A and an abstract normative constraint set Γ, or $\langle \Gamma, A, s\text{-}ob \rangle$-*obedient*, if at any point in the computation there exists an $a \in A$ such that it holds that for every $\langle \{a\}, \gamma \rangle \in \Gamma_{\{a\}}$ it is the case that if $\langle \{a\}, \gamma \rangle$ is enabled, then $\langle \{a\}, \gamma \rangle$ is not taken.
3. **Neglectful obedience** A computation λ with respect to a set of agents A and an abstract normative constraint set Γ is always neglectful obedient, or $\langle \Gamma, A, \top \rangle$-*obedient*. In other words, every computation is by definition neglectful obedient.
4. **Selective/Total individual disobedience** A computation λ is *selective individual disobedient* with respect to a set of agents A and an abstract normative constraint set Γ, or $\langle \Gamma, A, s\text{-}dob \rangle$-*obedient*, if at any point in the computation there exists an $a \in A$ such that it holds that there exists $\langle \{a\}, \gamma \rangle \in \Gamma_{\{a\}}$ such that if $\langle \{a\}, \gamma \rangle$ is enabled, then $\langle \{a\}, \gamma \rangle$ is taken. A computation λ is *total individual disobedient* with respect to a set of agents A and an abstract normative constraint set Γ, or $\langle \Gamma, A, t\text{-}dob \rangle$-*obedient*, if at any point in the computation for all $a \in A$ it holds that there exists $\langle \{a\}, \gamma \rangle \in \Gamma_{\{a\}}$ such that if $\langle \{a\}, \gamma \rangle$ is enabled, then $\langle \{a\}, \gamma \rangle$ is taken.
5. **Coalitional disobedience** A computation λ is *coalitional disobedient* with respect to a set of agents A and an abstract normative constraint set Γ, or $\langle \Gamma, A, c\text{-}dob \rangle$-*obedient*, if at any point in the computation for every $\langle A', \gamma \rangle \in \Gamma_A^-$ it is the case that if $\langle A', \gamma \rangle$ is enabled in λ, then $\langle A', \gamma \rangle$ is taken.

From these definitions, we see that the following holds for obediences: Given that a computation is $\langle \Gamma, A, c\text{-}ob \rangle$-obedient, it is also $\langle \Gamma, A, t\text{-}ob \rangle$-obedient. Given that it is $\langle \Gamma, A, t\text{-}ob \rangle$-obedient, it is also $\langle \Gamma, A, s\text{-}ob \rangle$-obedient and given that it is $\langle \Gamma, A, s\text{-}ob \rangle$-obedient, it is also $\langle \Gamma, A, \top \rangle$-obedient. For the disobediences a similar result holds: $\langle \Gamma, A, c\text{-}dob \rangle$-obedience implies $\langle \Gamma, A, t\text{-}dob \rangle$-obedience, $\langle \Gamma, A, t\text{-}dob \rangle$-obedience implies $\langle \Gamma, A, s\text{-}dob \rangle$-obedience and $\langle \Gamma, A, s\text{-}dob \rangle$-obedience implies $\langle \Gamma, A, \top \rangle$-obedience. An easy way of verifying this is by considering that each step in these implications allows for more possible state-transitions to take place in a computation.

4. an-ATL SEMANTICS

In this section we will give the semantics of our new logic an-ATL; ATL extended in order to reason under the presence of these abstract norms. We will first define the notion of obedience to the level of strategies. We define the set Ω as the set of obedience types denoted by a single literal, thus we have that $\Omega = \{c\text{-}ob, t\text{-}ob, s\text{-}ob, \top, s\text{-}dob, t\text{-}dob\}$. Given a game structure $S = \langle k, Q, \Pi, \pi, Ac, \delta \rangle$, an abstract normative constraint set Γ, a state q and an obedience type $\omega \in \Omega$, we say that the strategy set S_A for players A is $\langle \Gamma, A, \omega \rangle$-obedient if it holds that for every computation $\lambda \in out(q, S_A)$, λ is $\langle \Gamma, A, \omega \rangle$-obedient. In our semantics, an obedience assump-

tion is of the form $\langle \omega, \omega' \rangle$, where $\omega, \omega' \in \Omega$. The satisfaction relation $S, \Gamma, q \models \phi$ in our new semantics replaces the following cases in ATL semantics:

- $S, \Gamma, q \models \langle\!\langle A_{\langle \omega,\omega' \rangle} \rangle\!\rangle \bigcirc \varphi$ iff there exists a $\langle \Gamma, A, \omega \rangle$-obedient strategy set S_A for A such that for every $\langle \Gamma, \Sigma \backslash A, \omega' \rangle$-obedient computation $\lambda \in out(q, S_A)$ it holds that $S, \Gamma, \lambda[1] \models \varphi$.
- $S, \Gamma, q \models \langle\!\langle A_{\langle \omega,\omega' \rangle} \rangle\!\rangle \Box \varphi$ iff there exists a $\langle \Gamma, A, \omega \rangle$-obedient strategy set S_A for A such that for every $\langle \Gamma, \Sigma \backslash A, \omega' \rangle$-obedient computation $\lambda \in out(q, S_A)$ and all positions $i \geq 0$ it holds that $S, \Gamma, \lambda[i] \models \varphi$.
- $S, \Gamma, q \models \langle\!\langle A_{\langle \omega,\omega' \rangle} \rangle\!\rangle \varphi_1 \mathcal{U} \varphi_2$ iff there exists a $\langle \Gamma, A, \omega \rangle$-obedient strategy set S_A for A such that for every $\langle \Gamma, \Sigma \backslash A, \omega' \rangle$-obedient computation $\lambda \in out(q, S_A)$ there exists a position $i \geq 0$ such that for all positions $0 \leq j < i$ it holds that $S, \Gamma, \lambda[j] \models \varphi_1$ and $S, \Gamma, \lambda[i] \models \varphi_2$.

The formula $\langle\!\langle A_{\langle \omega,\omega' \rangle} \rangle\!\rangle \varphi$ should intuitively be read as "coalition A has an ω-obedient strategy in order to enforce φ if the remaining agents $\Sigma \backslash A$ played in accordance with an ω'-obedient computation". Moreover, we write $[\![A_{\langle \omega,\omega' \rangle}]\!] \bigcirc \varphi$ for $\neg \langle\!\langle A_{\langle \omega,\omega' \rangle} \rangle\!\rangle \bigcirc \neg \varphi$ and $[\![A_{\langle \omega,\omega' \rangle}]\!] \Box \varphi$ for $\neg \langle\!\langle A_{\langle \omega,\omega' \rangle} \rangle\!\rangle \Diamond \neg \varphi$ (where $\Diamond \varphi \equiv \top \mathcal{U} \varphi$; similar abbreviations can be defined for the dual of the $\mathcal{U}$ operator). The formula $[\![A_{\langle \omega,\omega' \rangle}]\!] \varphi$ should be read as "coalition A does not have a ω-obedient strategy in order to avoid φ if the remaining agents played in accordance with an ω'-obedient computation". Slightly similar to the result found in ATL, we have the following validity (notice the reversal of the obedience assumption tuple):

$$\models \langle\!\langle A_{\langle \omega,\omega' \rangle} \rangle\!\rangle \varphi \rightarrow [\![(\Sigma \backslash A)_{\langle \omega',\omega \rangle}]\!] \varphi$$

Moreover, the following validities hold, where β can either be replaced with "ob" or "dob":

1. $\models \langle\!\langle A_{\langle c\text{-}\beta,\omega \rangle} \rangle\!\rangle \varphi \rightarrow \langle\!\langle A_{\langle t\text{-}\beta,\omega \rangle} \rangle\!\rangle \varphi$
2. $\models \langle\!\langle A_{\langle t\text{-}\beta,\omega \rangle} \rangle\!\rangle \varphi \rightarrow \langle\!\langle A_{\langle s\text{-}\beta,\omega \rangle} \rangle\!\rangle \varphi$
3. $\models \langle\!\langle A_{\langle s\text{-}\beta,\omega \rangle} \rangle\!\rangle \varphi \rightarrow \langle\!\langle A_{\langle \top,\omega \rangle} \rangle\!\rangle \varphi$

Intuitively, what these formula's state is that a coalition, when having an obedient strategy to guarantee φ (under a certain assumption that the remaining agents play in accordance with an ω'-obedient computation), they also have a "less restrictive" obedient strategy (here, with "less restrictive" we mean that they are allowed to violate more normative constraints) to guarantee the same (and vice versa for disobedient strategies). This is as expected, since we have a richer pool of strategies to choose from when we have less restriction to cope with. Conversely, when reasoning about the other players, the implication works the other way around. In this case, we have the following validities (where again β can either be replaced with "ob" or "dob"):

1. $\models \langle\!\langle A_{\langle \omega,\top \rangle} \rangle\!\rangle \varphi \rightarrow \langle\!\langle A_{\langle \omega,s\text{-}\beta \rangle} \rangle\!\rangle \varphi$
2. $\models \langle\!\langle A_{\langle \omega,s\text{-}\beta \rangle} \rangle\!\rangle \varphi \rightarrow \langle\!\langle A_{\langle \omega,t\text{-}\beta \rangle} \rangle\!\rangle \varphi$
3. $\models \langle\!\langle A_{\langle \omega,t\text{-}\beta \rangle} \rangle\!\rangle \varphi \rightarrow \langle\!\langle A_{\langle \omega,c\text{-}\beta \rangle} \rangle\!\rangle \varphi$

This intuitively means that a coalition, when having a certain strategy to guarantee φ under a certain assumption that the other agents play in accordance with a certain obedient computation, they also have a strategy to guarantee the same under a "more restrictive" obedience assumption of the computation they move along to (where "more restrictive" means that there are less normative constraints which can be violated). This is again as expected, since we have to take into account less computations; namely we do not have to consider the computations where a normative constraint was enabled and taken which was previously not considered.

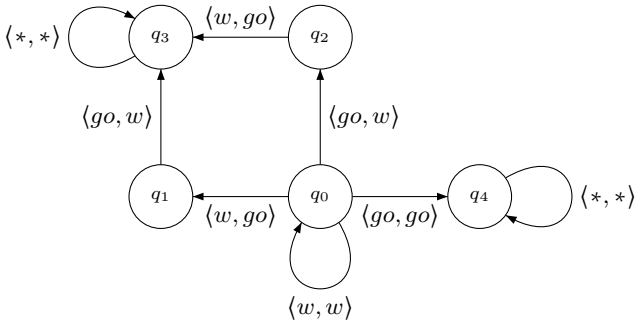


Figure 1: CGS train example.

4.1 Example

In this example we consider a scenario where there are two trains, each controlled by one agent, at the opposite ends of a tunnel. The tunnel only has room for one train, and the agents initially have 2 actions available: 'wait' and 'go'. If the agents decide to go simultaneously, the trains will crash, if they wait simultaneously nothing will happen and if one goes and the other waits, they can both successfully make it to the end of the tunnel without crashing. The CGS belonging to this scenario is shown in Figure 1. Let us denote the first agent as a_1 and the second agent as a_2. Moreover, we assume we have only two atomic propositions, 'crash' and 'no_crash', the former which is only true in q_4 and the latter only in q_3. Right of the bat, we can conclude that the following holds for every possible Γ:

$$M, \Gamma, q_0 \not\models \langle\!\langle a_{1\langle \top,\top \rangle} \rangle\!\rangle \Diamond \text{no\_crash}$$

and

$$M, \Gamma, q_0 \not\models \langle\!\langle a_{2\langle \top,\top \rangle} \rangle\!\rangle \Diamond \text{no\_crash}$$

In words, no agent individually has the ability to eventually bring about the fact that the trains make it through the tunnel without crashing. Moreover, the agents are collectively able to make the trains crash, displayed by the following formula (again for every possible Γ):

$$M, \Gamma, q_0 \models \langle\!\langle \{a_1, a_2\}_{\langle \top,\top \rangle} \rangle\!\rangle \Diamond \text{crash}$$

We will now construct a normative constraint set Γ in such a way that it brings about the following:

1. It is normatively demotivated for both agents that they enter the tunnel simultaneously.
2. It is normatively demotivated for the second agent, a_2, that he waits when the other train has not gone through the tunnel yet.

As it will turn out, the second normative constraint is sufficient for agent 1 to conclude that he has the ability to bring about the fact they will eventually reach the end of the tunnel safely under certain obedience assumptions of the other agent. However, from the perspective of agent 2, this constraint is still not sufficient: even if he adopts an obedient strategy, it is still not guaranteed that the trains will not

crash.

We can formalize the above mentioned constraints as abstract normative constraints as follows. We construct $\Gamma = \{\langle\{a_1, a_2\}, \gamma\rangle, \langle\{a_2\}, \gamma'\rangle\}$, such that:

- $\gamma(a_1, q_0) = \{\text{go}\}$, $\gamma(a_2, q_0) = \{\text{go}\}$; and
- $\gamma'(a_2, q_0) = \{\text{w}\}$

In this example, we are first going to reason about the abilities of agent one, a_1. The following an-ATL formula is valid:

$$M, \Gamma, q_0 \vDash \langle\!\langle a_{1\langle \top, t\text{-}ob\rangle} \rangle\!\rangle \Diamond \text{no\_crash}$$

In words, our first agent now has a strategy to eventually bring about the fact that the train makes it through the tunnel without crashing under the assumption that a_2 plays according to an obedient computation (in this case, he only has to take into account the normative constraint associated with γ'). To see this, observe that we do not consider any computations with transitions from q_0 to q_0 any more (and from q_0 to q_2, but this is beyond the point), thus disallowing the scenario where both agents wait for each other. The strategy for agent 1 is then to first wait (since he knows that agent 2 will not wait), and then go. Interestingly enough, agent 1 also has a strategy to eventually safely reach the end of the tunnel under the assumption that the other agent plays in accordance with disobedient computation:

$$M, \Gamma, q_0 \vDash \langle\!\langle a_{1\langle \top, t\text{-}dob\rangle} \rangle\!\rangle \Diamond \text{no\_crash}$$

To easily see this, observe that we do not consider any computations with transitions from q_0 to q_4 (and from q_0 to q_1, but again is beyond the point), thus the strategy where agent 1 immediately picks 'go' will ensure that both agents will eventually make it to the end of the tunnel safely.

Let us now reason about the abilities of agent a_2. We can see that a_2, as opposed to a_1, can not obediently bring about that both agents will reach the end of the tunnel safely:

$$M, \Gamma, q_0 \nvDash \langle\!\langle a_{2\langle t\text{-}ob, \top\rangle} \rangle\!\rangle \Diamond \text{no\_crash}$$

To see this, selecting the action 'go' in q_0 will not guarantee that we will not end up in q_4. However, interestingly enough, agent 2 does have the ability to disobediently bring about the fact that the ability of agent 1 to reach the end of the tunnel safely (under assumption that agent 2 plays in such a way that his normative constraints are not violated) is not lost:

$$M, \Gamma, q_0 \vDash \langle\!\langle a_{2\langle t\text{-}dob, \top\rangle} \rangle\!\rangle \bigcirc \langle\!\langle a_{1\langle \top, t\text{-}ob\rangle} \rangle\!\rangle \Diamond \text{no\_crash}$$

To see this, observe that the action 'wait' for agent 2 in q_0 will result in either q_0 or q_2, both in which the ability of agent 1 to safely reach the end of the tunnel, under assumption that agent 2 obeys his normative constraints, is retained. However, in case agent 2 plays an obedient strategy, this is not guaranteed, as seen by the following validity:

$$M, \Gamma, q_0 \nvDash \langle\!\langle a_{2\langle t\text{-}ob, \top\rangle} \rangle\!\rangle \bigcirc \langle\!\langle a_{1\langle \top, t\text{-}ob\rangle} \rangle\!\rangle \Diamond \text{no\_crash}$$

The reason for this is that agent 2 can not guarantee with an obedient strategy (action 'go' in q_0) that he will not end up in q_4. We could say that agent 2 is faced with a dilemma: he can play obedient, possibly wasting the ability of the other agent to reach the end of the tunnel safely if the other player assumed obedience over the normative constraints of agent 2, or play disobedient, preserving the former mentioned ability but possibly ending up in the same state again.

Let us finally reason about the abilities of the coalition of agents a_1 and a_2. We already saw that the coalition of agents have the ability to bring about the crashing of the trains. However, even if the individual agents play a totally individual obedient strategy (or a selective individual obedient strategy for that matter), the agents can still select a strategy that can make the trains crash, illustrated by the following validities:

$$M, \Gamma, q_0 \vDash \langle\!\langle \{a_1, a_2\}_{\langle t\text{-}ob, \top\rangle} \rangle\!\rangle \Diamond \text{crash}$$

and

$$M, \Gamma, q_0 \vDash \langle\!\langle \{a_1, a_2\}_{\langle s\text{-}ob, \top\rangle} \rangle\!\rangle \Diamond \text{crash}$$

The reason for this is because these obedience assumptions apply only to the agents on an individual level, and thus we only have to take into account the abstract normative constraint associated with a_2. On the coalitional level, the agents indeed do not have an coalitional obedient strategy to bring about the fact that eventually a crash will occur:

$$M, \Gamma, q_0 \nvDash \langle\!\langle \{a_1, a_2\}_{\langle c\text{-}ob, \top\rangle} \rangle\!\rangle \Diamond \text{crash}$$

Moreover, to illustrate how strong some obedience assumptions can be, the following formula holds for every possible φ:

$$M, \Gamma, q_0 \nvDash \langle\!\langle \{a_1, a_2\}_{\langle c\text{-}dob, \top\rangle} \rangle\!\rangle \varphi$$

The reason for this is because there does not exist a joint strategy that violates both abstract normative constraints simultaneously. This illustrates an important distinction with ATL, since ATL assumes that there is always a strategy available for any (sub)coalition of agents at any moment in time. This also inherently means that checking the validity of an an-ATL formula of the form $\langle\!\langle A_{\langle \omega, \omega' \rangle} \rangle\!\rangle \varphi$ in a model M cannot be reduced to checking a formula of the form $\langle\!\langle A \rangle\!\rangle \varphi$ in a transformed model M' with removed edges, the reason simply being that M' might not be an actual valid concurrent game structure any more.

5. MODEL CHECKING

The problem of determining whether a formula in an-ATL is satisfied at a certain state reduces to the application of model checking to the concurrent game structure. The model checking problem for transition systems is discussed in [5], and in [3] model checking for ATL is discussed.

We define the extended game structure

$$S^F = \langle k, Q^F, \Pi^F, \pi^F, Ac^F, \delta^F \rangle$$

as follows:

- $Q^F = \{\langle \bot, q\rangle \mid q \in Q\} \cup$
  $\{\langle q', q\rangle \mid q', q \in Q \text{ and } q \text{ is a successor of } q' \text{ in } Q\}$.
- $\Pi^F = \Pi \cup (\Gamma \times \{enabled, taken\})$.
- For all $\langle \bot, q\rangle \in Q^F$ we have $\pi^F(\langle \bot, q\rangle) = \pi(q)$.
  For all $\langle q', q\rangle \in Q^F$ we have $\pi^F(\langle q', q\rangle) = \pi(q) \cup$
  $\{\langle\langle A, \gamma\rangle, enabled\rangle \mid \forall a \in A : \gamma(a, q') \neq \varnothing\} \cup$
  $\{\langle\langle A, \gamma\rangle, taken\rangle \mid$ exists $\langle \alpha_1, ..., \alpha_k\rangle \in D(q')$ such that $\forall a \in A : \alpha_a \in \gamma(a, q')$ and $\delta(q', \langle \alpha_1, ..., \alpha_k\rangle) = q\}$

- For all $a \in \Sigma$ and all $\langle \bot, q\rangle, \langle q', q\rangle \in Q^F$ we have $Ac^F(a, \langle \bot, q\rangle) = Ac^F(a, \langle q', q\rangle) = Ac(a, q)$.
- For all $\langle \bot, q\rangle, \langle q', q\rangle \in Q^F$ and all $\langle \alpha_1, ..., \alpha_k\rangle \in D(q)$ we have $\delta^F(\langle \bot, q\rangle, \langle \alpha_1, ..., \alpha_k\rangle) = \delta^F(\langle q', q\rangle, \langle \alpha_1, ..., \alpha_k\rangle) = \langle q, \delta(q, \langle \alpha_1, ..., \alpha_k\rangle)\rangle$.

Let f^Γ_A be a function that maps, given a coalition A and abstract normative constraint set Γ, each obedience assumption in Ω to a propositional formula with variables $\Gamma \times \{enabled, taken\}$. Thus we have $f^\Gamma_A : \Omega \mapsto \mathcal{L}_{prop}(\Gamma \times \{enabled, taken\})$. We define the function as follows (notice that we have written $\langle \varphi, e\rangle$ and $\langle \varphi, t\rangle$ as shorthand for $\langle \varphi, enabled\rangle$ and $\langle \varphi, taken\rangle$ respectively):

1. $$f^\Gamma_A(c\text{-}ob) = \bigwedge_{\langle A,\gamma\rangle \in \Gamma^-_A} (\langle\langle A, \gamma\rangle, e\rangle \rightarrow \neg\langle\langle A, \gamma\rangle, t\rangle)$$
2. $$f^\Gamma_A(t\text{-}ob) = \bigwedge_{a \in A}, \bigwedge_{\langle\{a\},\gamma\rangle \in \Gamma_{\{a\}}} (\langle\langle\{a\}, \gamma\rangle, e\rangle \rightarrow \neg\langle\langle\{a\}, \gamma\rangle, t\rangle)$$
3. $$f^\Gamma_A(s\text{-}ob) = \bigvee_{a \in A}, \bigwedge_{\langle\{a\},\gamma\rangle \in \Gamma_{\{a\}}} (\langle\langle\{a\}, \gamma\rangle, e\rangle \rightarrow \neg\langle\langle\{a\}, \gamma\rangle, t\rangle)$$
4. $$f^\Gamma_A(\top) = \top$$
5. $$f^\Gamma_A(s\text{-}dob) = \bigvee_{a \in A}, \bigvee_{\langle\{a\},\gamma\rangle \in \Gamma_{\{a\}}} (\langle\langle\{a\}, \gamma\rangle, e\rangle \rightarrow \langle\langle\{a\}, \gamma\rangle, t\rangle)$$
6. $$f^\Gamma_A(t\text{-}dob) = \bigwedge_{a \in A}, \bigvee_{\langle\{a\},\gamma\rangle \in \Gamma_{\{a\}}} (\langle\langle\{a\}, \gamma\rangle, e\rangle \rightarrow \langle\langle\{a\}, \gamma\rangle, t\rangle)$$
7. $$f^\Gamma_A(c\text{-}dob) = \bigwedge_{\langle A,\gamma\rangle \in \Gamma^-_A} (\langle\langle A, \gamma\rangle, e\rangle \rightarrow \langle\langle A, \gamma\rangle, t\rangle)$$

We then claim that evaluating a formula of the form $S, \Gamma, q \vDash \langle\!\langle A_{\langle\omega,\omega'\rangle}\rangle\!\rangle\varphi$ amounts to model checking an ATL* formula in the extended game structure.

PROPOSITION 1. *$S, \Gamma, q \vDash \langle\!\langle A_{\langle\omega,\omega'\rangle}\rangle\!\rangle\varphi$ holds if and only if:*

$$S^F, \langle \bot, q\rangle \vDash \langle\!\langle A\rangle\!\rangle(\Box f^\Gamma_A(\omega) \wedge (\Box f^\Gamma_{(\Sigma\setminus\Gamma)}(\omega') \rightarrow \varphi))$$

Although this is an ATL* formula, the model checking complexity can still be done in efficient time (polynomial in the number of transitions, the size of the abstract normative constraint set and the length of the an-ATL formula). The exact details of this result are not relevant for this paper, but we can give a basic intuition behind this finding. Consider a game structure S with m transitions and an abstract normative constraint set Γ of size g. We start out by constructing $S^F = \langle k, Q^F, \Pi^F, \pi^F, Ac^F, \delta^F\rangle$ from S. Notice that if S has m transitions, S^F has $O(m)$ states and $O(m)$ transitions. Now, the interesting cases arise when we want to check a sub-formula of the form $\langle\!\langle A_{\langle\omega,\omega'\rangle}\rangle\!\rangle \Box\, \varphi$ or $\langle\!\langle A_{\langle\omega,\omega'\rangle}\rangle\!\rangle\varphi_1\mathcal{U}\varphi_2$. The idea now is that, when looking for states satisfying these conditions, we can just encode the winning and losing conditions in the game structure itself. We do this by adding two states to the game structure S^F, an "always winning" and an " always losing" state. The "always winning" state q_W makes the goal formula true forever, the "always losing" state q_L makes the goal formula false forever regardless of the goal formula $\Box\varphi$ or $\varphi_1\mathcal{U}\varphi_2$. Notice that the states q_W and q_L do not follow the usual definitions of a Concurrent Game Structures; for example the formula $\Box\bot$ is still true at q_W even though there would not exist a valuation for q_W that makes this possible. However for the sake of model-checking this does not matter. Now consider we are model checking a formula $\langle\!\langle A_{\langle\omega,\omega'\rangle}\rangle\!\rangle\psi$ (where $\psi = \Box\theta$ or $\psi = \theta_1\mathcal{U}\theta_2$). We proceed as follows: for each state $q \in Q^F$, if it holds that $q \vDash \neg f^\Gamma_A(\omega)$, then redirect all incoming transitions to state q_L, if it holds that $q \vDash \neg f^\Gamma_{(\Sigma\setminus A)}(\omega') \wedge f^\Gamma_A(\omega)$, then redirect all incoming transitions to q_W. This routine can be done in efficient time, and ensures that if A can only select $\langle\Gamma, A, \omega\rangle$-obedient transitions to make the goal formula true (else it would end up in q_L) and ensures that if $\Sigma\setminus A$ selects a non $\langle\Gamma, \Sigma\setminus A, \omega'\rangle$-obedient transition the goal formula is by default satisfied. Now we can just proceed with "normal" model checking, which as shown in [3], can be done in time proportional to the number of transitions in the concurrent game structure, which is $O(m)$.

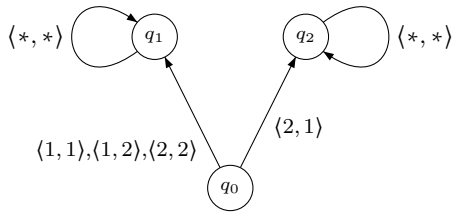


Figure 2: CGS with 2 agents.

6. SELF-SUPPORTING SETS

When we introduced the notion of abstract normative constraints, we saw that that it is still possible that an abstract normative constraint of the form $\langle A, \gamma\rangle$ is taken at a certain state in the model, even though the agents in A might not have selected the exact actions prescribed by γ. Thus, even though it seems they were selecting an action in order to avoid a violation, they still ended up in a situation where this is not the case. In these special circumstances it is the case that the agents do not have the power to autonomously avoid a violation. Autonomy is a key facet within the (multi)agent paradigm and can play a major role for the agents to decide whether they want to participate in the multiagent system or comply with the given norms, so we devote this last section to identify and classify the circumstances in which a normative constraint set limits the autonomy of (coalitions of) agents.

Let us first consider the concurrent game structure shown in Figure 2. Moreover, let us consider the normative constraint set $\Gamma = \{\langle\{a_1\}, \gamma\rangle\}$, where $\gamma(a_1, q_0) = \{1\}$, we see that agent 2, while being in state q_0, has the ability to enforce agent 1 into a violation. Namely, we see that agent 2 can select action 2, which causes agent 1 to end up in q_1 regardless of the action he chooses. Since there exists an action for agent 1 that disallows going from state q_0 to q_1 (namely action 1), even by selecting action 2 the agent can not avoid violating the constraint. This brings us to the notion of

self-supporting constraint sets. Intuitively, self-supporting means that if a normative constraint is targeted towards coalition A, the agents in this coalition have (in some way) "control" over whether or not they will violate this particular constraint. However, as we will see in this section, multiple gradations of "control" can be given. We start out with a weak form of self-supporting, called *weakly self-supporting*.

Definition 1 (Weakly Self-supporting). *Given a concurrent game structure S and abstract normative constraint set Γ, we say that Γ is weakly self-supporting with respect to S if and only if it holds that for every coalition $A \subseteq \Sigma$ and at any state $q \in Q$ there is no strategy available to A in order to force the remaining players $\Sigma \backslash A$ into a non-$\langle \Gamma, \Sigma \backslash A, c\text{-}ob \rangle$-obedient computation.*

We see that in our previous example this was not the case since agent 2 could force agent 1 into a violation by selecting action 2. The following proposition shows how we can identify weakly self-supporting constraint sets using our new an-ATL logic.

Proposition 2 (Weakly Self-supporting). *Given a concurrent game structure S and abstract normative constraint set Γ, Γ is weakly self-supporting with respect to S if and only if for all $A \subseteq \Sigma$ it holds that:*

$$S, \Gamma \vDash [\![A_{\langle \top, c\text{-}ob \rangle}]\!] \bigcirc \top$$

Let us give an intuition about why this proposition holds. Observe that, for a given A, the formula $[\![A_{\langle \top, c\text{-}ob \rangle}]\!] \bigcirc \top$ is equal to $\neg \langle\!\langle A_{\langle \top, c\text{-}ob \rangle} \rangle\!\rangle \bigcirc \bot$. Now let us suppose that $\langle\!\langle A_{\langle \top, c\text{-}ob \rangle} \rangle\!\rangle \bigcirc \bot$ holds at state q. This means that there exists a strategy for A, let us call this S_A, such that if coalition $\Sigma \backslash A$ behaved according to a $\langle \Gamma, \Sigma \backslash A, c\text{-}ob \rangle$-obedient computation, $\bigcirc \bot$ would be true. However, the latter can never be true for any computation, thus we must conclude that every computation in $out(q, S_A)$ is not $\langle \Gamma, \Sigma \backslash A, c\text{-}ob \rangle$-obedient, which implies that Γ is not weakly self-supporting with respect to S. A similar reasoning can be applied for the other way around.

Suppose for now we have a weakly self-supporting constraint set Γ with respect to S. Now, even though there exists no coalition that can enforce the other players into a non collective obedient computation, it does not mean that every coalition has a collective obedient strategy available to them. Consider the CGS shown in Figure 3, with again the normative constraint set $\Gamma = \{\langle \{a_1\}, \gamma \rangle\}$, where $\gamma(a_1, q_0) = \{1\}$. We see that Γ is weakly self-supporting with respect to S, since it is not possible for agent 2 to force agent 1 into a non-$\langle \Gamma, \{a_1\}, c\text{-}ob \rangle$-obedient computation. However, it is not the case that agent 1 has a collective obedient strategy available to him as both action 2 and 3 might bring him into state q_1. This brings up a more stronger notion of self-supporting constraint sets, namely those in which each coalition always has a collective obedient strategy available to them. If this is the case, we say that a normative constraint set is *strongly self-supporting* with respect to a concurrent game structure.

Definition 2 (Strongly Self-supporting). *Given a concurrent game structure S and abstract normative constraint set Γ, we say that Γ is strongly self-supporting with respect to S if and only if it holds that for every coalition $A \subseteq \Sigma$ there is at any state $q \in Q$ a collective obedient strategy available to them.*

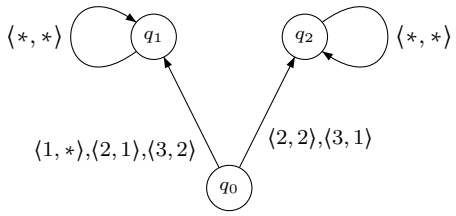


Figure 3: CGS with 2 agents.

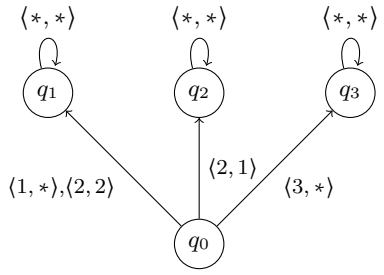


Figure 4: Another CGS with 2 agents.

Just like in the case of weakly self-supporting constraint sets, we can identify when a normative constraint set is strongly self-supporting with respect to a concurrent game structure with the use of our an-ATL logic.

Proposition 3 (Strongly Self-supporting). *Given a concurrent game structure S and abstract normative constraint set Γ, we say that Γ is strongly self-supporting with respect to S if and only if for all $A \subseteq \Sigma$ it holds that:*

$$S, \Gamma \vDash \langle\!\langle A_{\langle c\text{-}ob, \top \rangle} \rangle\!\rangle \bigcirc \top$$

We now claim that if a normative constraint set is strongly self-supporting, it is also weakly self-supporting. This is not hard to verify because, as we have already seen in Section 4, we have the result that $S, \Gamma, q \vDash \langle\!\langle A_{\langle \omega, \omega' \rangle} \rangle\!\rangle \varphi$ implies $S, \Gamma, q \vDash [\![(\Sigma \backslash A)_{\langle \omega', \omega \rangle}]\!] \varphi$, and thus that for all $A \subseteq \Sigma$, $S, \Gamma \vDash \langle\!\langle A_{\langle c\text{-}ob, \top \rangle} \rangle\!\rangle \bigcirc \top$ implies for all $A \subseteq \Sigma$, $S, \Gamma \vDash [\![A_{\langle \top, c\text{-}ob \rangle}]\!] \bigcirc \top$ (but not the other way around).

Again suppose we have a strongly self-supporting constraint set Γ with respect to S. An example can be seen in Figure 4 with again the normative constraint set $\Gamma = \{\langle \{a_1\}, \gamma \rangle\}$, where $\gamma(a_1, q_0) = \{1\}$. Although there is a collective obedient strategy available for agent 1 in q_0, namely selecting at this state action 3, there is still an action available to him which is not in the constraint set but can causes him to violate a normative constraint, i.e., if agent 1 selects action 2 in q_0, there is still a possibility that agent 2 selects action 2. This gives rise to yet another notion of self-supporting constraint sets, namely that of *unconditional self-supporting normative constraint sets*. Intuitively, if this is the case it means that only the (joint) actions normatively demotivated by the constraint set Γ will result in a violation. Thus, "unconditional" does not mean that the status of whether or not a constraint set is self-supporting does not rely on the game structure itself, it merely means that if it is established that a constraint set is unconditionally self-supporting, it is sufficient to only look at the constraint set in order to select a collective obedient strategy. To make

this more formal, let us introduce the notion of an A-move. An A-move is a function c_q^A that maps each player $a \in A$ to an action for that player, thus we have $c_q^A(a) \in Ac(a,q)$. We write $C(A,q)$ for the set of all A-moves c_q^A for coalition A at state q. Now we define $C_\Gamma(A,q)$ as the set of all A-moves such that it holds that:

$$C_\Gamma(A,q) = \{c_q^A \in C(A,q) \mid \forall \langle \gamma, A' \rangle \in \Gamma_A^- : allows(\langle \gamma, A' \rangle, c_q^A)\}$$

where: $allows(\langle \gamma, A' \rangle, c_q^A) =$

$$(\forall a \in A' : \gamma(a,q) \neq \varnothing) \Rightarrow \bigvee_{a \in A'} c_q^A(a) \notin \gamma(a,q)$$

In words, the set $C_\Gamma(A,q)$ contain all A-moves for coalition A which are not normatively demotivated by a constraint in Γ_A^-. We can now give the formal definition of unconditional self-supporting constraint sets.

Definition 3 (Unconditional Self-supporting). . *Given a concurrent game structure $S = \langle k, Q, \Pi, \pi, Ac, \delta \rangle$, we say that an abstract normative constraint set Γ is unconditional self-supporting with respect to S if and only if for all $A \subseteq \Sigma$ it holds that for all states $q \in Q$ it is the case that $C_\Gamma(A,q)$ is non-empty and for every A-move $c_q^A \in C_\Gamma(A,q)$, it holds that by playing it, all of the normative constraints $\langle A', \gamma \rangle \in \Gamma_A^-$ will be either not enabled or not taken.*

Note that an unconditional self-supporting normative constraint set is (by definition) also strongly self-supporting since we demanded $C_\Gamma(A,q)$ to be non-empty. Now given a concurrent game structure $S = \langle k, Q, \Pi, \pi, Ac, \delta \rangle$ and an abstract normative constraint set Γ, we can define out_q as a function from a set of A-moves to the set of all possible states the agents can arrive in by playing such an A-move.

$$out_q(C(A,q)) = \{\delta(q, d_q) \in Q \mid d_q \in g(c_q^A) \text{ and } c_q^A \in C(A,q)\}$$

where

$$g(c_q^A) = \{\langle \alpha_1, ..., \alpha_k \rangle \in D(q) \mid \forall a \in A : c_q^A(a) = \alpha_a\}$$

Using this definition, the following proposition now states how we can verify when an abstract normative constraint set is unconditional self-supporting with respect to a game structure.

Proposition 4 (Unconditional Self-supporting). *A normative constraint set Γ with respect to a concurrent game structure $S = \langle k, Q, \Pi, \pi, Ac, \delta \rangle$ is unconditional self-supporting iff $\forall q \in Q$ and $\forall A \subseteq \Sigma : out_q(C_\Gamma(A,q)) \neq \varnothing$ and $out_q(C_\Gamma(A,q)) \cap out_q(C(A,q) \backslash C_\Gamma(A,q)) = \varnothing$.*

To see why this proposition holds, note that it states that the states reachable from an A-move in $C_\Gamma(A,q)$ (all the A-moves which are not normatively demotivated by the constraints in Γ_A^-) and the states reachable from $C(A,q) \backslash C_\Gamma(A,q)$ (all the A-moves which are normatively demotivated by the constraints in Γ_A^-) must be disjoint. If this is not the case, then there exists an $A-move$ in $C_\Gamma(A,q)$ which by playing it would result in one of the constraints in Γ_A^- to be enabled and taken, thus not unconditionally self-supporting. A similar reasoning can be applied for the other way around.

What we have seen in this section is that it is possible to characterize and identify multiple levels of "control" the agents have over the ability to adhere to the normative constraints. As we already argued, identifying when a normative constraint set is not (weakly/strongly/unconditional) self-supporting with respect to a concurrent game structure may play a crucial role for the agents in order to decide whether to participate in the system, since they restrict the autonomy of the agents in some way.

7. DISCUSSION AND FUTURE RESEARCH

In this paper we have developed a model of normative systems that allows reasoning about agents' (normative) abilities under a multitude of compliance assumptions. This can be both crucial in the design and validation of these systems. To do this, we introduced the notion of abstract normative constraints and we developed an extension of Alternating Temporal Logic, an-ATL, to allow reasoning about the abilities of (coalitions of) agents under different compliance assumptions. Moreover, we showed that model-checking an-ATL formula's remains close to the complexity of model-checking standard ATL. In the last part of the paper, we discussed the notion of self-supporting constraint sets and explained its relation to autonomy of agents. In particular, we explained that if a constraint set is self-supporting, the agents have (in some way) control over whether they can avoid a violation.

There are multiple ways to extend this line of research. Firstly, it would be very interesting to consider more compliance assumptions. As can be seen in Section 5, given a normative constraint set Γ, our logic allows us to create arbitrary complex obedience assumptions in the language $\mathcal{L}_{prop}(\Gamma \times \{enabled, taken\})$. Moreover, it would be interesting to incorporate goals and preferences of agents and see how they relate to the obedience behaviour of the other agents. Finally, the question what "good coalitions" for agents are in order to not violate any of the norms must be answered. Perhaps this question can be related to the topic of coalitional game theory.

8. REFERENCES

[1] T. Ågotnes, W. van der Hoek, and M. Woolridge. Robust normative systems and a logic of norm compliance. *Logic journal of the IGPL*, 18:4–30, 2010.

[2] T. Ågotnes, M. Wooldridge, and W. van der Hoek. Normative system games. In *AAMAS'07*, pages 876–883, 2007.

[3] R. Alur, T. A. Henzinger, and O. Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49(5):672–713, 2002.

[4] N. Bulling and M. Dastani. Verifying normative behaviour via normative mechanism design. In *IJCAI'11*, pages 103–108, 2011.

[5] E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, Cambridge, 1999. ISBN 0-262-03270-8.

[6] Y. Moses and M. Tennenholtz. Artificial social systems. *Computers and AI*, 14(6):533 – 562, 1995.

[7] Y. Shoham and M. Tennenholtz. On the synthesis of useful social laws for artificial agent societies. In *AAAI'92*, pages 276–281, 1992.

[8] W. van der Hoek, M. Roberts, and M. Wooldridge. Social laws in alternating time: effectiveness, feasibility, and synthesis. *Synthese*, 156(1):1–19, 2007.

[9] M. Woolridge and W. van der Hoek. On obligations and normative ability. *Journal of Applied Logic*, 3:396–420, 2005.

Session 5B
Teamwork II

Leading Ad Hoc Agents in Joint Action Settings with Multiple Teammates*

Noa Agmon and Peter Stone
Department of Computer Science
The University of Texas at Austin
{agmon,pstone}@cs.utexas.edu

ABSTRACT

The growing use of autonomous agents in practice may require agents to cooperate as a team in situations where they have limited prior knowledge about one another, cannot communicate directly, or do not share the same world models. These situations raise the need to design *ad hoc* team members, i.e., agents that will be able to cooperate without coordination in order to reach an optimal team behavior. This paper considers the problem of leading N-agent teams by an agent toward their optimal joint utility, where the agents compute their next actions based only on their most recent observations of their teammates' actions. We show that compared to previous results in two-agent teams, in larger teams the agent might not be able to lead the team to the action with maximal joint utility, thus its optimal strategy is to lead the team to the best possible *reachable* cycle of joint actions. We describe a graphical model of the problem and a polynomial time algorithm for solving it. We then consider other variations of the problem, including leading teams of agents where they base their actions on longer history of past observations, leading a team by more than one ad hoc agent, and leading a teammate while the ad hoc agent is uncertain of its behavior.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Agent Cooperation::Teamwork, coalition formation, coordination ; Economic paradigms::Game theory (cooperative and non-cooperative)

1. INTRODUCTION

Teams of agents have been studied for more than two decades, where the general assumption is that the agents coordinate their actions to increase the team's performance. The growing popularity of agents in domains such as e-commerce, has raised the need for cooperation between agents that are not necessarily programmed similarly, and might not have the same communication protocols or world models. Nevertheless, these agents might be required to perform as a coordinated team. When designing such systems, one cannot assume the team members engage in a known team strategy, but each agent must adjust to the current circumstances while adopting a strategy that aims to optimize the team's utility. Such systems are called *Ad-Hoc Teams*.

In many cases, such as robotic teamwork, it might be impossible to change the design of agents in a team. This work attempts to provide theoretical results (model and solution, and bound on existence of a solution) for the possible influence of a new added agent (one or more) on the team performance. Consider the case where several robots are deployed on Mars; you designed them (thus know their behavior), but once they are there - you cannot re-code them. Suppose that as time passes, you have more knowledge about the world. Will it be worthwhile to send a new robot to change the team behavior to a new, improved, one? If so - how should it do so? These questions motivate our research, and this paper makes progress towards answering them.

Specifically, we consider the problem of leading a team to the optimal joint action. In this problem, the team members do not communicate explicitly, but are assumed to choose their actions based on their observations of their teammates' previous actions (one or more), i.e., the agents behave as *best response agents*.[1] The problem is represented as a simultaneous repeated game. The ultimate goal is to have all team members act in a way that will maximize the joint utility of the team. Assume we design a team member that joins the team, the *ad hoc team member*. Our goal is, therefore, to determine the optimal strategy for the ad hoc team member such that it will lead the team to their optimal joint action while minimizing the system's cost while doing so.

This problem was introduced by Stone *et al.* [11] for systems of two agents: one ad hoc agent, and one best response agent. They describe an algorithm for determining the optimal strategy for the ad hoc agent that leads, in minimal cost, the best response agent to perform the action yielding optimal joint utility. In this paper we consider the more general problem of leading N-agent teams, $N \geq 2$, toward their optimal joint utility. In such systems, the possible

*This work has taken place in the Learning Agents Research Group (LARG) at UT Austin. LARG research is supported in part by NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030).

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

[1]We consider best response agents for simplicity, however our results can equally be applied to the more general case of agents that base their decisions on a fixed history window of the ad hoc agents' past actions, rather than on their own previous actions.

influence of one agent on the team is relatively limited compared to the two-agent teams. As a result, we show that in N-agent teams, the optimal joint utility might not be reachable, regardless of the actions of our agent. In such cases, our agent's optimal strategy is to lead the team to the best possible *reachable* joint utility, with minimal cost.

In order to find the optimal strategy for the ad hoc team player, we describe a graphical model of the possible joint set of actions integrating the possible transitions between the system's states (agents' joint action), and the resulting costs of those transitions. Using this model, we first determine the set of joint actions resulting in maximal joint utility, and find the lowest cost path from the initial joint action of the agents to this optimal set of joint actions. We then consider other variations of the problem, and evaluate them using the suggested graphical model. These variations include leading best-response agents with memory size greater than one, leading a team by a team of coordinated ad hoc agents, and leading a two-agent team by an ad hoc agent where uncertainty exists on the behavior of its teammate.

2. PROBLEM DESCRIPTION

We consider the problem of leading a team of best response players by one ad hoc team member towards the optimal joint utility of the team. In this section we describe the general problem, as well as notations that will be used throughout the paper.

The problem of finding a policy for leading team members to the optimal joint utility was introduced in [11] for a team of two agents, A and B, where agent A is the ad hoc agent and agent B is the best response agent. Agent A was designed to lead agent B to perform an action that will result in the optimal joint utility, denoted by m^*. This is done *without using explicit communication or prior coordination*, where agent B chooses an action that maximizes the joint utility based on its observation of agent A's previous action (but with both players having knowledge of the game). This is designed as a simultaneous repeated game, i.e., the players choose their actions simultaneously, where current actions influence the future decisions of the players.

The assumption is that agent B's behavior is known to agent A, but is fixed and unchangeable. This represents one of the simplest cases of ad hoc teamwork, where there is no uncertainty about behaviors. Nevertheless, it poses some interesting challenges, as shown previously in [11], and reinforced in this paper. Relaxation of this assumption is discussed in Section 6.

Agent A has x possible actions $\{a_0, \ldots, a_{x-1}\}$, and agent B has y possible actions $\{b_0, \ldots, b_{y-1}\}$. The team's utility is represented by an $x \times y$ payoff matrix M, where an entry $M(i,j) \in M$ is the joint utility when A performs action a_i and B performs b_j. The *cost* of a joint action $(a_i, b_j), 0 \leq i \leq x-1, 0 \leq j \leq y-1$, denoted by $C(a_i, b_j)$, is defined as $m^* - M(i,j)$, i.e., the distance from the optimal joint utility. The system is initialized in the joint action (a_0, b_0).

It can be assumed, without loss of generality, that m^* is the joint utility obtained when A performs action a_{x-1} and B performs b_{y-1}. Therefore m^* is necessarily reachable from (a_0, b_0) in at most two stages: A picks a_{x-1} and B will adjust in the next stage and choose action b_{y-1}, thus a possible sequence to the optimal joint action m^* is $\langle (a_0, b_0), (a_{x-1}, b_0), (a_{x-1}, b_{y-1}) \rangle$, after which A and B will continue performing actions a_{x-1} and b_{y-1} (respectively). However, this might not be the only possible sequence, and moreover - there could be a sequence of joint actions leading to m^* that has lower cost. The question answered by Stone *et al.* [11] was, therefore, how to reach m^* with minimal cost. They describe a dynamic programming algorithm for finding the optimal solution in polynomial time. Their solution is based on the knowledge of the longest possible optimal sequence, bounding the depth of the recursive algorithm.

In this paper we consider the more general case of leading N-agent teams, $N \geq 2$, by one ad hoc team player. We do so by first examining the problem of leading three-agent teams, and then describe the generalization to N agent teams.

The three-agent team consists of agent A - the ad hoc team member, and the best response agents B and C. The set of actions available for the agents is $\{a_0, \ldots, a_{x-1}\}$, $\{b_0, \ldots, b_{y-1}\}$ and $\{c_0, \ldots, c_{z-1}\}$ for agents A, B, and C, respectively. The payoff matrix of the team is a 3-D matrix M, that can be conveniently written as x matrices of size $y \times z$, $M_0, \ldots, M_{x-1}$ (see Figure 1), where entry (b_i, c_j) in matrix M_k, denoted by $M_k(i,j)$, $(0 \leq k \leq x-1,\ 0 \leq i \leq y-1,\ 0 \leq j \leq z-1)$, is the payoff of the system when agent A performs action a_k, B performs b_i and C performs c_j. Denote the maximal joint payoff in the system by m^*, and assume, without loss of generality, that the agents initially perform actions (a_0, b_0, c_0). Similarly to the two-agent case, the agents do not coordinate explicitly, but agents B and C are assumed to choose their next move according to their current observation of their teammates' actions. Therefore the next action of agent B (denoted by BR_B) is based on its current observation of the actions of agents A and C, and similarly the next action of C (denoted by BR_C) is based on the actions of A and B. Formally, $BR_B(a_i, c_k) = \text{argmax}_{0 \leq j \leq y-1}\{M_i(j,k)\}$ (similarly for BR_C). Therefore if the current actions of the agents are (a_i, b_j, c_k), then the next joint action would be $(a_{i'}, BR_B(a_i, c_k), BR_C(a_i, b_j))$ for $0 \leq i, i' \leq x-1$, $0 \leq j \leq y-1$ and $0 \leq k \leq z-1$.

	c_0	c_1
b_0	5	2
b_1	1	4

	c_0	c_1
b_0	8	6
b_1	4	**20**

Figure 1: An example for unreachable $m^* = (a_1, b_1, c_1)$

Compared to the two-agent team, in a three-agent team the control of agent A on the world is relatively limited. Specifically, there are cases in which m^* remains unreachable, regardless of the actions of agent A. An example for such a case is given in Figure 1. In this example, $x = y = z = 2$. According to these payoff matrices, $BR_B(a_i, c_0) = b_0$ $BR_C(a_i, b_0) = c_0, i \in \{0, 1\}$, thus agents B and C will continue to choose actions (b_0, c_0) in both matrices, and A will not be able to lead them to joint utility of $m^* = M_1(1,1) = 20$. Therefore the goal of agent A is to lead the team to the *best possible reachable* joint action or cycle of joint actions. In this example, A will choose action a_1, leading to the maximal possible payoff of 8, and all agent will continue choosing the same action yielding maximal possible joint utility.

Definition A *Steady Cycle* is a sequence of t joint actions $s_0, s_1, \ldots, s_{t-1}$ such that if $s_l = (a_i, b_j, c_k)$, then $s_{l+1} = (a_{i'}, BR_B(a_i, c_k), BR_C(a_i, b_j))$, $(0 \leq l \leq t-1,\ 0 \leq i; i' \leq x-1,\ 0 \leq j \leq y-1,\ 0 \leq k \leq z-1)$, and $s_t = s_0$, i.e., the sequence is cyclic. The *Optimal Steady Cycle*, denoted by OSC, is a steady cycle with minimal average cost, i.e, $1/t \times \sum_{i=1}^{t} C(s_i)$ is minimal.

Note that if m^* is reachable, the optimal steady cycle consists of only the joint action yielding payoff m^*.

3. LEADING A TEAM BY A SINGLE AGENT

In this section we examine the problem of leading a team by a single agent, initially concentrating on three-agent teams. We describe a graphical model for representing the system, and a polynomial time algorithm for determining the optimal possible set of joint actions and how to reach it with minimal cost to the team. We later (Subsection 3.4) generalize the representation and solution to an $N-$agent teams.

3.1 Problem Definition

The three-player team consists of three agents: agent A, our designed ad-hoc team player, and agents B and C, which are the original team players.

We define the 3-***Player Lead to Best Response Problem*** (3LBR) as follows.

Given a three-agent team, A, B, C, where agent A is an ad-hoc team player and agents B and C are best response players, and a 3-D payoff matrix representing the team payoff for every joint action of the agents, determine the set of actions of agent A that will lead the team to the optimal steady cycle reachable from (a_0, b_0, c_0) in minimal cost.

3.2 Graphical Representation

In this section we describe a graphical model of the state space, used to find an optimal solution to the 3LBR problem. We create a graph $G = (V, E)$, where V includes of all possible joint actions, i.e., each vertex $v_{ijk} \in V$ corresponds to a set of joint actions (a_i, b_j, c_k) $(0 \leq i \leq x-1, 0 \leq j \leq y-1, 0 \leq i \leq z-1)$. The directed edges in E are defined as follows: an edge $e = (v_{ijk}, v_{i'j'k'}) \in E$ $\forall i', 0 \leq i' \leq x-1$, if $j' = BR_B(a_i, c_k)$ and $k' = BR_C(a_i, b_j)$. In words, an edge is added where it is possible to move from one set of joint actions to the other—either by A repeating the same action $(a_i = a_{i'})$ or by it switching to another action $a_i \neq a_{i'}$. The cost of an edge $e = (v_{ijk}, v_{i'j'k'})$ is set to $m^* - M_{i'}(b'_j, c'_k)$. The total number of vertices in G is xyz, and the number of edges is $x \times |V| = x^2yz$.

Figure 2 illustrates two sets of payoff matrices and their corresponding graphical representations. On the right, m^* is reachable, hence the optimal steady cycle is of size $t = 1$ and includes only v_{111}. The optimal path to the optimal steady cycle is the shortest path (corresponding to the path with lowest cost) between v_{000} to v_{111}, which is $v_{000}, v_{101}, v_{111}$, meaning that the minimal cost sequence is $\langle (a_0, b_0, c_0), (a_1, b_0, c_1), (a_1, b_1, c_1) \rangle$ with a total minimal cost of 21 (there is a "startup cost" of 15 for the first play, that is added to all paths from vertex 000, as indicated by the incoming edge to that vertex). The dashed lines represent the transitions which are determined by A's choices if it changes its action, leading to a change in M_i. The solid lines represent the outcome if A did not change its action.

3.3 Algorithm for Solving the 3LBR Problem

The solution to the 3LBR problem, described in Algorithm 1, is divided into two stages:

1. Find the optimal reachable steady cycle.
2. Find the sequence of actions for agent A that leads the team to the optimal steady cycle with minimal cost.

In order to find the optimal reachable steady cycle, we first remove all vertices that do not belong to the connected component that includes v_{000}. This can be done by a simple BFS tour starting at v_{000} (linear in the graph size). Finding the optimal steady cycle corresponds to finding the Minimum Cycle Mean (introduced by [8]), that can be computed using dynamic programming in time $\mathcal{O}(|V| \times |E|) = \mathcal{O}(x^3y^2z^2)$.

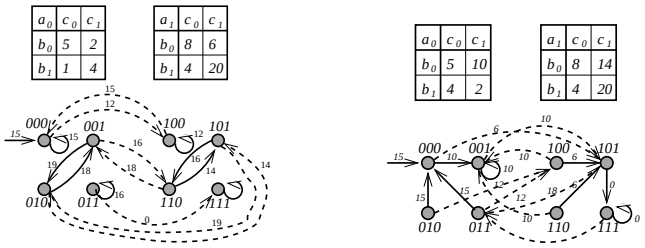


Figure 2: An example for the graphical representation of the state transition. On the left - the representation of the the payoff matrix from Figure 1 where m^* is unreachable, and on the right a case in which m^* is reachable.

Finding the sequence of actions taken by agent A that will lead the team to the optimal steady cycle with minimal cost is equivalent to finding the shortest path from v_{000} to any vertex in the cycle yielding the minimum cycle mean of G. Recall that the number of vertices in the cycle yielding the minimum cycle mean of G is denoted by t. Therefore finding the shortest path from v_{000} to one of the vertices of that cycle can be done by Dijkstra's algorithm, resulting in time complexity of $\mathcal{O}(t|E|\log|V|) = \mathcal{O}(tx^2yz\log(xyz))$. The total time complexity of the algorithm is, therefore, $\mathcal{O}(tx^2yz\log(xyz) + x^3y^2z^2)$.

Comparing the time complexity of our algorithm to the algorithm presented by Stone *et al.* [11] for two-player games, we note that finding the optimal sequence of joint actions for a two-player game is a special case of the three-player game in which $z = 1$, and the optimal reachable steady cycle is known to be v_{110}. Thus the time complexity of finding the optimal sequence using our algorithm is $\mathcal{O}(x^2y\log(xy))$, compared to $\mathcal{O}(x^2y)$ of the algorithm described in Stone *et al.* [11]. However, as they have shown, there is no point in returning to a set of joint actions in this scenario, thus while constructing the graph, edges closing a cycle will not be added, yielding a directed acyclic graph (DAG). In DAGs, the shortest path can be found in $\mathcal{O}(|E| + |V|)$ (using topological ordering), therefore the time complexity is similar to the one described in [11], i.e., $\mathcal{O}(x^2y)$.

Algorithm 1 Algorithm Lead3Team(M)

```
Create graph G = (V, E) as follows
for 0 ≤ i ≤ x − 1 ; 0 ≤ j ≤ y − 1 ; 0 ≤ k ≤ z − 1 do
    Add v_{i,j,k} to V
end for
for 0 ≤ i ≤ x − 1 ; 0 ≤ j ≤ y − 1 ; 0 ≤ k ≤ z − 1 do
    for 0 ≤ i' ≤ x − 1 do
        Add e = (v_{i,j,k}, v_{i',BR_B(i,k),BR_C(i,j)}) to E
        Set w(e) = m* − M(i', BR_B(i,k), BR_C(i,j))
    end for
end for
Compute G' ⊆ G by a BFS tour on G starting from v_{0,0,0}
Compute the optimal steady cycle OSC ⊆ G' = {v^0, ..., v^{k−1}}
(Minimum Cycle Mean)
P ← argmin_{0≤i<k} {Shortest path from v_{0,0,0} to v^i ∈ OSC}
```

3.4 Leading N-agent teams

Generalizing 3LBR, we define the *N-Player Lead to Best Response* Problem (NLBR) as follows.

Let $\{a^0, \ldots, a^{N-1}\}$ be a team of N agents, where agent a_0

is an ad-hoc team player. The set of actions for each team member a^i $(0 \leq i \leq N-1)$ is $\{a^i_0, a^i_1, \ldots, a^i_{r_i}\}$, and we are given an N-D payoff matrix M representing the team payoff for every combination of actions of the agents. Determine the set of actions of agent a_0 that will lead the team to the optimal steady cycle reachable from $(a^0_0, a^1_0, \ldots, a^{N-1}_0)$ in minimal cost.

In order to generalize the solution to the 3LBR problem to the NLBR problem, it is necessary to define its graphical representation. Once the graphical model is set, finding the optimal solution to the problem becomes similar to the three-agent case, i.e., we find the optimal steady cycle, then we find the shortest path to that cycle. The main difference from the three-agent case lies, therefore, in the creation of the graph, which in turn affects the time complexity.

The graph $G = (V, E)$ is built similarly to the 3-player game, where $v_{i_0 i_1 \ldots i_{N-1}} \in V$ for each set of joint actions $(a^0_{i_0}, a^1_{i_1}, \ldots, a^{N-1}_{i_{N-1}})$ corresponding to an entry in the payoff matrix M_{i_0}, and $e = (v_{i_0 i_1 \ldots i_{N-1}}, v_{i'_0 i'_1 \ldots i'_{N-1}}) \in E$ if $\forall 1 \leq j \leq N-1$, $a^j_{i'_j} = BR_j(a^0_{i_i}, \ldots, a^{j-1}_{i_{j-1}}, a^{j+1}_{i_{j+1}}, \ldots, a^{N-1}_{i_{N-1}})$, $\forall 0 \leq i' \leq r_0 - 1$.

The number of vertices in G is $\prod_{i=0}^{N-1} r_i$, and the number of edges is $r_0 \prod_{i=0}^{N-1} r_i$, leading to a time complexity of $\mathcal{O}(tr_0^2 \prod_{i=1}^{N-1} r_i \log(\prod_{i=0}^{N-1} r_i) + r_0 \prod_{i=0}^{N-1} r_i^2)$ for solving the NLBR problem (t is the length of the optimal steady cycle).

4. LEADING A TEAM WITH MEMORY> 1

Until this point, we have assumed that the teammates optimize their choices based on their most recent observation (best response). We now consider the case in which team members have memory greater than one, i.e., each agent computes its best response to the maximum likelihood distribution corresponding to the last mem joint actions it observed. We describe the analysis for three-agent teams; the solution for the general N-agent team follows directly.

Denote the number of times agent A, B and C performed action a_i, b_j and c_k during the previous set of mem joint actions by N^a_i, N^b_j and N^c_k, correspondingly. Formally, for a three-agent team, the best response of agents B and C are defined (BR_B and BR_C, correspondingly) as follows:
$BR_B = \text{argmax}_{\{0 \leq l \leq y-1\}} \{\sum_{i=0}^{x-1} \frac{N^a_i}{mem} \sum_{k=0}^{z-1} \frac{N^c_k}{mem} M_i(l,k)\}$
(BR_C is defined similarly). Let $BR_B(s_1, \ldots, s_{\mathsf{mem}})$ ($BR_C(s_1, \ldots, s_{\mathsf{mem}})$) be the best response of agent B (C) based on the last mem observed joint actions $s_1, \ldots, s_{\mathsf{mem}}$.

The graph representation $G = (v, e)$ in case mem > 1 is as follows. A vertex $v \in V$ represents a global state which includes a sequence of mem consecutively executed joint actions, $\{s_0, \ldots, s_{\mathsf{mem}}\}$. An edge $e = (v, u) \in E$ exists from a vertex $v = \{s_0, \ldots, s_{\mathsf{mem}}\}$ to vertex $u = \{s'_0, \ldots, s'_{mem}\}$ if $\forall 0 \leq l \leq \mathsf{mem} - 2; \forall 0 \leq i \leq x-1$, $s'_l = s_{l+1}$, $s'_{\mathsf{mem}} = \{a_i, BR_B(s_0, \ldots, s_{\mathsf{mem}}), BR_C(s_0, \ldots, s_{\mathsf{mem}})\}$.

As shown by Stone *et al.* [11], even if m^* was played once, it does not necessarily mean that the system will stay there. Specifically, it could be the case that the team was lead to m^*, however the next best response of some agent (one or more) would be to switch actions. This was denoted as *unstable states*. Assume the joint action yielding m^* is (a^*, b^*, c^*). As a result, we define the terminal vertex of the system to be $\langle (a^*, b^*, c^*), \ldots, (a^*, b^*, c^*) \rangle$ (mem times). Clearly, this vertex is stable.

4.1 On the time complexity of handling high memory

Finding the optimal steady cycle and the optimal path to that cycle in case the team members have memory size greater than one can be computed similarly to the solution to the 3LBR and the NLBR problems (Algorithm Lead3Team). In order to determine the time complexity of reaching an optimal solution, it is necessary to calculate the size of the graph representation, i.e., $|V|$ and $|E|$. The number of combinations of mem sets of joint actions is $|V|^{\mathsf{mem}}$. However, not all combinations of sets of joint actions are feasible (the system cannot reach every vertex from every vertex, but only x vertices from each vertex), hence the upper bound on the number of vertices is $xyz \times x^{\mathsf{mem}-1}$, i.e., $x^{\mathsf{mem}} yz$ (exponential in mem). The number of edges from each vertex remains x, hence the total number of edges is $x^{\mathsf{mem}+1} yz$.

This provides an *upper bound* on the time complexity of reaching a solution with mem ≥ 2. However, we would like to examine whether this bound is tight or not, i.e., can we guarantee that the time complexity will practically be lower than that? We do so by pursuing two different directions. First, we check whether there is a connection between the relevant graph size (connected component that includes the initial vertex) with teammates having memory of size mem $- 1$ and the relevant graph size when their memory is mem. For example, if the connected component only got smaller as memory increased, then we could bound the graph size by the size of the connected component when mem=1. Second, we check whether we can efficiently bound the possible length of the optimal path from the initial joint action to the optimal (cycle of) joint action(s). If so, that would allow us to restrict the construction of our state space to states within that maximal length from the initial state. Unfortunately, the investigations in both directions are not promising. We show that there is no connection between the size of the relevant connected component as the memory size grows (it could increase or decrease). We also show that even in the simplest case of $N = 2$ and mem $= 2$ determining the maximal size of an optimal path from $v_{0,0}$ to m^* is $\mathcal{NP}$-Hard.

4.1.1 Graph size as memory size changes

We investigated the influence of the number of vertices in the connected component in case mem $= 1$ to the number of components when mem $= 2$ in order to determine a tight bound on the number of vertices we need to explore as the memory grows. Unfortunately, we have shown that there is no relation between the number of vertices in the connected components between different memory sizes. In this section, we describe two opposite cases: one in which the connected component grows, and one in which it becomes smaller.

We show, by the following example, that the connected component originating at $v = (0, 0, 0)$ with mem $= 1$ can grow as mem grows to include vertices corresponding to joint actions that were unreachable with smaller memory size. Moreover, Figure 3 shows a case in which with mem $= 1$ m^* is unreachable, yet with mem $= 2$ it becomes reachable. On the other hand, as shown in Figure 4, the number of reachable joint actions may decrease, possibly causing m^* to become unreachable.

These two examples demonstrate that a tight bound on the number of vertices that need to be explored as memory grows does not exist, i.e., it is not sufficient to explore only

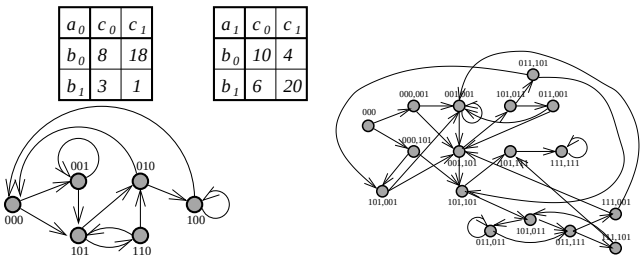


Figure 3: An example for a case where m^* was unreachable for mem $= 1$ (left), and became reachable with mem $= 2$ (right).

the main connected component of mem $= 1$, but it is necessary to explore the entire main connected component in the current memory model.

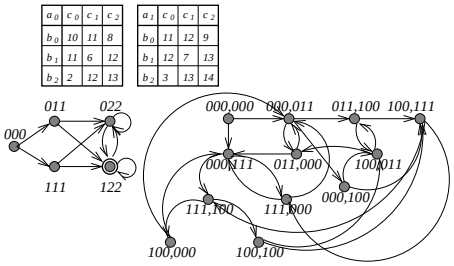


Figure 4: An example for a case where m^* was reachable for mem $= 1$ (left), and became unreachable with mem $= 2$ (right).

4.1.2 $\mathcal{NP}$-Hardness of maximal optimal length determination

In order to evaluate whether the graph size can be reduced to guarantee a more realistic upper bound on the time complexity that is not exponential in the graph size, we examined the possibility of limiting the construction of the graph based on the maximal possible length of an optimal path from the initial vertex to the optimal steady cycle. In [11] it was shown that in two agent teams, consisting of one ad hoc team member (agent A) and one best-response team member (agent B), if mem $= 1$ then the maximal size of an optimal path is at most $\min\{x, y\}$. However, we prove here that even in this simple case where m^* is known to be reachable (i.e., the optimal steady cycle is known in advance), determining the maximal size of an optimal path is $\mathcal{NP}$-Hard when agent B has mem ≥ 2 (note that this was assumed in [11], yet was not proven there).[2]

Denote the maximal length of an optimal path starting at (a_0, b_0) to m^* by S^* (note that since it is the two agent game, such a path always exists).

THEOREM 1. *In the two-agent case, finding S^* when* mem ≥ 2 *is $\mathcal{NP}$-Hard.*

PROOF. The problem is proven to be NP-hard by a reduction from the Hamiltonian Path problem:Given an n-node unweighted, undirected graph G, an initial node and a destination node, is there a simple path from initial to destination of length n? That is, can we visit each node exactly once? This decision problem is known to be $\mathcal{NP}$-Complete [4].

[2]We thank ML for his help in formalizing the proof

We will show that if it were possible to find S^* for a given matrix M with agent B's mem > 1 in polynomial time, then it would also be possible to find a Hamiltonian path in polynomial time. To do so, we assume that we are given a graph $G = (V, E)$ such that $|V| = n$. We construct a matrix M in a particular way such that if there is a path through the matrix of cost no more than a target value of $n*(n^4-1)$, then there is a Hamiltonian Path in graph G. Note that, as required, the construction of the matrix can be done in time polynomial in all the relevant variables.

Let agent B's mem $= n$. We construct the joint payoff matrix M as follows.

- Agent A has $(n-1)*n+2$ actions. The first action is "start", and agent B's memory is initialized to n copies of that action. Each of the next $(n-1)*n$ actions represents a combination (i,t) of a vertex $v_i \in V(G)$ and a time step $t \geq 2$. M's payoffs will be constructed so that if the sequence satisfying the maximum cost requirement in M (if any) includes action (i,t), then the corresponding Hamiltonian path passes through v_i on time step t. Finally, there is a "done" action to be taken at the end of the path.
- Agent B has n^2+n+1 actions. The first n^2 actions are similar to agent A's: one for each combination of $v_j \in V(G)$ and $t \geq 1$. If the satisfying sequence through M includes agent B taking action (j,t), then the Hamiltonian path visits v_j at time t. The next n actions are designed as "trap" actions which agent B will be induced to play if agent A ever plays two actions corresponding to the same node in the graph: actions (i,s) and (i,t). There is one trap action for each vertex, called action j. Finally, the last action is the "done" action to be played at the end of the sequence.
- M's payoffs are constructed as follows, with the actions named as indicated in the bullets above. The initial vertex in the Hamiltonian path (the one visited on time step 1) is called "initial."

a)	$M[(i,t+1),(j,t)]$	$=$	1	if $(v_i, v_j) \in E(G)$
b)	$M[(i,t+1),(j,t)]$	$=$	$-n^5$	if $(v_i, v_j) \notin E(G)$
c)	$M[(i,t),(i,t)]$	$=$	tn	
d)	$M[(i,t),(j,s)]$	$=$	$-n^5$	if $t \geq s$
e)	$M[(i,t),(j,s)]$	$=$	0	if $t < s$
f)	$M[(i,t),i]$	$=$	$tn - \frac{1}{3n}$	
g)	$M[(i,t),j]$	$=$	0	
h)	$M[(i,t),\text{done}]$	$=$	0	
i)	$M[\text{start},(\text{initial},1)]$	$=$	1	
j)	$M[\text{start},\text{initial}]$	$=$	$\frac{1}{2}$	
k)	$M[\text{start},\text{done}]$	$=$	$-n^4$	
l)	$M[\text{start},j]$	$=$	0	$\forall$ action $j \notin$ {initial,done}
k)	$M[\text{done},(j,n)]$	$=$	1	
l)	$M[\text{done},(j,t)]$	$=$	$-n^5$	if $t < n$
m)	$M[\text{done},\text{done}]$	$=$	n^4	

Following a path through the matrix that corresponds to a Hamiltonian path (if one existed) would give payoffs of 1 at every step until reaching m^* (n^4) and staying there forever. Thus the cost of the n-step path would be $n*(n^4-1)$.

As there is no positive payoff in the matrix greater than n^2, any path longer than n steps must have cost of at least $(n+1)(n^4-n^2) = n^5+n^4-n^3-n^2 > n^5-n = n*(n^4-1)$. In other words, if there is a path through the matrix corresponding to a Hamiltonian path in the graph, then any longer path through M must have higher cost.

Furthermore, the matrix is carefully constructed such that any diversion from the path corresponding to a Hamiltonian

path either will get a payoff of $-n^5$ on at least one step (which by itself makes the target cost impossible to reach), will prevent us from getting one of the 1's, or else will make it so that the path to (done,done) will require more than n total steps. In particular, if agent A ever takes two actions that lead agent B to select a trap action, then agent B will not take a different action until the $n+1$st step after the first action that led to the trap, causing the path to (done,done) to be at least $n+2$ steps long. Therefore, if we could find the optimal sequence through any matrix in polynomial time, then we could use this ability to also solve the Hamiltonian path problem, concluding our proof. □

5. LEADING A TEAM BY A TEAM

Until now, we have considered the case of one ad-hoc agent leading a team of best response agents towards the optimal set of joint actions. However, it could be possible to deploy a *team* of coordinated ad-hoc agents to lead the best response agents. The two interesting questions that arise here are: (a.) What is the time complexity of determining the optimal path? (b.) What is the influence of the addition of a new team member to the group with respect to the optimal steady cycle and the cost of the path towards it?

Also in this case we adopt the graphical model in order to find the optimal steady cycle, and the shortest path towards it. Let $\{a^0, \ldots, a^{N-1}\}$ be a team of N agents, where agents $a^0, \ldots a^{k-1}$ are the ad-hoc team players, and $a^k, \ldots, a^{N-1}$ are the best response team members. Each agent a^i has a set of possible actions $\{a_0^i, \ldots, a_{r_i}^i\}$. Therefore the number of possible joint actions (hence the number of vertices in the representing graph), similar to the $N-$agent teams discussed in Section 3.4, is $\prod_{i=0}^{N-1} r_i$. The difference between the $N-$agent teams led by one agent and the $N-$agent teams led by k agents lies in the edges in the graph. Formally, the graph $G = (V, E)$ representing the system is built as follows. A vertex $v_{i_0 i_1 \ldots i_{N-1}} \in V$ exists for each joint action $(a_{i_0}^0, \ldots, a_{i_{N-1}}^{N-1})$. An edge $e = (v_{i_0 \ldots i_{N-1}}, v_{i'_0 \ldots i'_{N-1}}) \in E$ if $\forall k \leq j \leq N-1$, $a_{i'_j}^j = BR_j(a_{i_i}^0, \ldots, a_{i_{j-1}}^{j-1}, a_{i_{j+1}}^{j+1}, \ldots, a_{i_{N-1}}^{N-1})$, $\forall 0 \leq i'_l \leq r_{l-1}; 0 \leq l \leq k-1$. Therefore the number of outgoing edges from each vertex is $\prod_{i=0}^{k-1} r_i$, since each ad hoc team member's choice of action influences the possible outcome, hence $|E| = \prod_{i=0}^{k-1} r_i^2 \prod_{j=k}^{N-1} r_j$. An example for a graphical representation of leading a team by more than one agent is shown in Figure 5.

Finding the optimal steady cycle is done on this graph as described previously. The time complexity of determining the optimal steady cycle is $\mathcal{O}(|E| \times |V|) = \mathcal{O}(\prod_{i=0}^{k-1} r_i^3 \prod_{j=k}^{N-1} r_i^2)$. Assuming the number of joint actions in the steady cycle is $t \geq 1$, finding the optimal path from the initial vertex to the optimal steady cycle is done in time $\mathcal{O}(t|E| \log(|V|)) = \mathcal{O}(t \prod_{i=0}^{k-1} r_i^2 \prod_{j=k}^{N-1} r_i \log(\prod_{i=0}^{N-1} r_i))$

Clearly, as more agents are involved in leading the team, their influence on the outcome (the optimal reachable steady cycle) is higher relative to the case of a single leading agent. For example, in Figure 1, if agents B and C were leading the team, then m^* becomes reachable. However, in the general case, having more than one leader still might not have the power to lead the system towards m^*. Consider, as an example, the case in which $k = 2$ and $N = 4$, i.e., two agents lead the other two agents towards the optimal steady cycle. The payoff matrix can be considered as $r_0 \times r_1$ matrices of size $r_2 \times r_3$. If each of these matrices is, for example, a replica

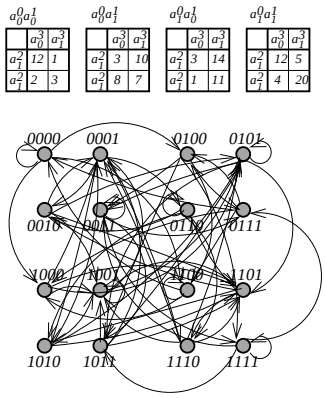


Figure 5: An example for the graphical representation where there is more than one ad hoc agent. The team has 4 agents: 2 ad hoc (a^0 and a^1) and 2 best response (a^2 and a^3), each having 2 actions.

of the matrix described in Figure 1, then agents a^0 and a^1 will never be able to lead the team to m^* (unless starting at m^*), as regardless of their actions, a^2 and a^3 will never jointly choose actions a_1^2 and a_1^3 (respectively).

This example leads to the following corollary:

COROLLARY 2. *In a team of N agents, for every $N \geq 3$, where at least two team members are best response agents, m^* may remain unreachable.*

6. LEADING WITH UNCERTAINTY

It is possible, perhaps likely, that our designed agent will be uncertain about the behaviors of its teammates. We consider in this section a two-agent team, agents A and B, where agent A is an ad hoc agent, and agent B can be either an ad hoc agent or a best response agent. Clearly, if B is an ad hoc agent and the agents are fully coordinated, then their choice of actions is well defined—both will choose together the joint action with utility m^*. Similarly, in larger teams where there is more than one ad hoc agent and they are coordinated, then the solution is similar to the one described in Section 5. However, uncertainty may arise when, for example, communication fails, and agent A has to decide which action to take in order to lead the team to m^* (recall that in the two-agent team case, m^* is always reachable). In cases where there exists a probability distribution over the possible identities of the teammate, the ad hoc agent can choose actions that result in a minimal cost path towards m^*, and also possibly choose actions that will reveal the true identity of the teammate (if necessary).

If B is a best response agent, then like in previous sections (and in [11]) it assumes that A will continue performing its previous action, and given that action it will choose an action maximizing the joint utility. If B is an ad hoc agent, we assume that it will act in one of two ways: it will either believe that A is an ad hoc player, thus will choose action b_{y-1} (assuming A will choose a_{x-1}, reaching m^* immediately), or it will assume that A is a best response player, and choose its actions appropriately. Modeling the identity of the other agent can be done recursively infinitely (what A believes B will do, that believes what A will do, and so on), however we are motivated by RMM [5] in modeling only a recursion of depth two (note that although RMM does recommend a recursive depth 2, it is usually shown to be beneficial to

model a bounded depth, and we leave the general discussion of depth in this setting to future work). Therefore we model three possible cases: (i) B is a best response agent, denoted by br ; (ii) B is an ad hoc agent, believing that A is an ad hoc agent, denoted by ah/ah; (iii) B is an ad hoc agent believing that A is a best response agent, denoted by ah/br.

In order to determine the best action for our agent A, we build a graph for each identity of the teammate. Note that since the vertices of the graphs are identical, we can use one graph with three different types of edges: E_i for br agent, E_{ii} for ah/ah agent and E_{iii} for the ah/br agent.

Generally, when uncertainty arises with respect to the environment (in our case the identity of the teammate), it is common to gain more information about the environment while exploring it. In our case, we need agent A to both gain information about B, and to allow agent B to reevaluate its belief about A. Specifically, if agent B is ah/br, we want it to realize that A is *not* a best response agent, thus allowing the team to reach m^* more efficiently. This happens if A diverts from the expected best response behavior. Note that if agent B is an ah/ah agent, it will choose to perform action b_{y-1} even after realizing that A did not perform a_{x-1}, since diverting from that action will necessarily cause the path to m^* to be longer (even in the case where A is best response, its next action would be a_{x-1}, leading to m^*).

As a first step we construct E_i, E_{ii} and E_{iii}. E_i is constructed as in Section 3. E_{ii} and E_{iii} can theoretically be complete graphs, however given the beliefs of B, most of the edges will not exist. Since B believes A to be an ad hoc agent, it will choose action b_{y-1}, thus E_{ii} includes x edges from v_{00} to $v_{i(y-1)}$, $\forall 0 \leq i \leq x-1$, and $x-1$ more edges from $v_{i(y-1)}$ to m^*. E_{iii} is constructed as follows. The only outgoing edge out of v_{00} is to some v_{ij} such that $a_i = BR_A(b_0)$, and $b_j \in SP_B(v_{00}, m^*)$, where SP_B denotes the shortest path from v_{00} to m^*, calculated by B (according to the weights and path that are calculated similarly to the description in Section 3). In addition, there are $x-1$ edges from v_{00} to v_{kj}, $0 \leq k \leq x-1$. From now on, for each vertex v, if there is no incoming edge from some vertex u such that v is the best response of u with respect to A's behavior as a best response agent, then we add an edge (v, m^*), otherwise the only edges added are from v to v_{ij}, $0 \leq i \leq x-1$, $b_j \in SP_B(v_{00}, m^*)$ (we omit the formal algorithm due to space constraints). Choosing the shortest path from v_{00} to m^* will determine whether it is more efficient to choose an edge that in the short term might not be beneficial, but will reveal enough information to the teammate that will force it to realize it is teamed with an ad hoc agent, making it choose a shorter path towards m^*.

Given the constructed edges, one can compute the shortest path along each set of edges, starting from v_{00} towards m^*. Each of these path has a total cost, and given a probability distribution over the possible identities of the teammates, the first action is chosen such that the expected cost is minimized. If a_{x-1} is chosen, then the path towards m^* is easily determined (regardless of the true identity of B). If it was not chosen, then A follows the path it chose, while making adjustments throughout the execution. Namely, if it chose an action that will teach B that it's an ad hoc agent, yet B acts as a best response (rather than ad hoc), it will find the shortest path towards m^* according to E_i (as it is clear that B is best response). Otherwise, it will follow the shortest path by E_{iii} towards m^*.

	b_0	b_1	b_2	b_3	b_4
a_0	10	15	0	0	0
a_1	0	16	17	0	5
a_2	0	0	18	19	15
a_3	0	0	0	20	21
a_4	0	0	0	0	22

Figure 6: Example for the advantage of relying on expected cost.

Clearly, if A assumes some identity of B and it is not mistaken, then this yields the best possible path towards m^*. Consider the example in Figure 6. If A prepares for ah/ah and is teamed with an ah/ah, the cost of path towards m^* is 0. If it expects B to be ah/br and B is indeed ah/br, the cost of the path is 6. Finally, if it expects B to be br and is indeed teamed with a br agent, then the cost is 12. Being wrong in the identity of the teammate is costly. Note that no matter if A expects a weak opponent and is teamed with a strong one or the opposite - the consequences (in terms of path cost) are high. In this example (considering a uniform probability distribution between the identities) assuming a br teammate has minimal expected cost of path towards m^*: assuming an ah/ah teammate has expected cost of 14.6, assuming an ah/br teammate has expected cost of 18.6 and br has expected cost of 14. Therefore, according to the algorithm, B should be assumed to be a br agent. This choice of action yields a *worst case* cost of mistaken identity of 17 (if B is actually ah/ah). On the other hand, if we would not have consulted the algorithm and chosen to believe that B is ah/ah, the worst cost of mistaken identity would be 22, and similarly ah/br yields worst cost of 28 upon mistaken identity. Note that generally even if the maximal cost of mistaken identity is not higher in the unchosen belief, still the expected cost is lower, thus in the average case, by using this algorithm, A minimizes its cost.

When leading a team with more than one agent where one of them might be an ad hoc agent, the identity of the teammate can be crucial, and determines whether m^* is reachable or not. This case is broad and complicated and is thus left out of the scope of this paper.

7. RELATED WORK

Stone *et al.* [10] introduced a formalism for *Ad-Hoc teamwork*, which deals with teamwork behavior without prior coordination. They raised a challenge *"To create an autonomous agent that is able to efficiently and robustly collaborate with previously unknown teammates on tasks to which they are all individually capable of contributing as team members"*. This paper answers one aspect of the challenge raised there, namely leading teams of agents, with no a-priori coordination and explicit communication to the optimal possible joint-utility, in a simultaneous-action setting.

Bowling and McCracken [1] suggested two techniques for incorporating a single agent into an unknown team of existing agents: adaptive and predictive. In their work, they are concerned with the task allocation of the agent (which role should it choose, and what is its teams' believed behavior), where their agent might adapt its behavior to what it observes by the team. Jones *et al.* [7] examined the problem of team formation and coordination without prior knowledge in the domain of *treasure hunt*. They considered a team composed of heterogenous robots, each with different capabilities required for various aspects of searching an unknown environment and extracting a hidden treasure. Their architecture was based on role selection using auctions. In contrast to these approaches, in our work we examine how our agent can influence the behavior of the team by leading

the team to an optimal behavior.

Stone and Kraus [12] considered the problem of ad hoc teamwork by two agents, agent A (also known as the teacher), and agent B in the k-armed bandit problem. The question they asked was: Assuming that agent B observes the actions of agent A and its consequences, what actions should agent A choose to do (which bandit to pull) in order to maximize the team's utility. It was shown that in some cases, agent A should act as a teacher to agent B by pulling a bandit that will not yield optimal immediate payoff, but will result in teaching agent B the optimal bandit it should pull. In our work we also control the actions of agent A, but the payoff is determined by the joint actions of the team players, not by individual actions of each teammate.

Han *et al.* [6] examined a closely related problem of controlling the collective behavior of self-organized multi-agent system by one agent. They consider self organized teams of physically interacting agents, concentrating on flocking of birds, where their goal is to design an agent, denoted as a *shill agent*, that will be able to gradually change the heading of the entire team to a desired heading. They evaluate the system in terms of physical capabilities of the shill agent and the team (velocity, initial heading) and provide theoretical and simulation results showing that it is possible, under some conditions, for one agent to change the heading of the entire team. Different from out approach, they do not consider game theoretic evaluation of the individual actions and their impact on the team behavior.

Young [13] introduced the notion of *adaptive games*, where N agents base their current decisions on a finite (small) horizon of observations in *repeated games*, and search for agents' actions yielding a stochastically stable equilibrium using shortest paths on graphs. In our work, we do not assume the agents play repeatedly (allowing to adjust to a strategy), but we aim to guarantee that our agent leads the team to the optimal possible joint action(s) while minimizing the cost paid by the team along the way.

Numerous research studies exist in the area of normal form games, where the agents' payoffs are described in a matrix (similar to our case) and depend on the chosen joint actions. In the normal form games framework, a related topic is the problem of learning the best strategy for a player in repeated games. Powers and Shoham [9] considered the problem of normal form games against an adaptive opponent with bounded memory. Chakraborty and Stone [2] examine optimal strategies against a memory bounded learning opponent. Cho and Kreps [3] introduced signaling games, a sequential Bayesian game in which a player chooses its actions based on messages transferred from a second player with unknown type. Our work is inherently different from these approaches, since in our case the agents are collaborating as a team, hence they aim to maximize the *joint* payoff and not the individual payoff, which raises different questions and challenges as for the optimality of the joint action and the way to reach this optimal joint action.

8. CONCLUSIONS AND FUTURE WORK

In this paper we examine the problem of leading a team of $N \geq 2$ agents by one or more ad hoc team members to the team's joint actions yielding optimal payoff. We show that it may not be possible to lead the team to the optimal joint action, and offer a graphical representation of the system's state and a polynomial time algorithm that determines the optimal *reachable* set of joint actions, and finds the path with minimal system cost to that set. We examine the case in which the team members base their next action on more than one previous joint action, describe an algorithm—using the same graphical representation—that calculates the optimal strategy for the ad hoc team member in time exponential in the teammates' memory size, and show that it is not likely that there exists an algorithm that solves the problem in better time complexity. We further use the graphical representation to consider the case in which the ad hoc agent is teamed with more than one ad hoc agent in coordination, and also the case in which it might be teamed with one agent with uncertain nature—ad hoc or best response.

There are various directions to be addressed as future work. First, the question of uncertainty can be examined not only in the identity of the agent, but also in many other aspects, such as the knowledge of the payoff matrix, namely, the ad hoc agent might not have full knowledge of the payoff matrix, but some incomplete knowledge or a probability distribution over possible values. Similar uncertainty may exist in the knowledge of the best response agents on the payoff matrix, which might result in nondeterministic choice of actions. Uncertainty may exist also in the type of teammates, acting not necessarily as best response agents, but adapting other (known or unknown) behaviors.

9. REFERENCES

[1] M. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *Proc. of AAAI'05*, pages 53–58, 2005.

[2] D. Chakraborty and P. Stone. Online multiagent learning against memory bounded adversaries. In *Machine Learning and Knowledge Discovery in Databases*, volume 5212 of *Lecture Notes in Artificial Intelligence*, pages 211–26, September 2008.

[3] I. Cho and D. M. Kreps. Signaling games and stable equilibria. *The Quarterly Journal of Economics*, 102(2):179–221, 1987.

[4] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[5] P. J. Gmytrasiewicz, E. H. Durfee, and D. K. Wehe. A decision-theoretic approach to coordinating multiagent interactions. In *IJCAI*, pages 62–68, 1991.

[6] J. Han, M. Li, and L. Guo. Soft control on collective behavior of a group of autonomous agents by a shill agent. *Systems Science and Complexity*, 19(1), 2006.

[7] E. Jones, B. Browning, M. Dias, B. Argall, M. Veloso, and A. Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *Proc. of ICRA'06*, pages 570 – 575, 2006.

[8] R. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Mathematics*, 23, 1978.

[9] R. Powers and Y. Shoham. Learning against opponents with bounded memory. In *Proc. of IJCAI'05*, pages 817–822, 2005.

[10] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proc. of AAAI'10*, 2010.

[11] P. Stone, G. A. Kaminka, and J. S. Rosenschein. Leading a best-response teammate in an ad hoc team. In *Agent-Mediated Electronic Commerce: Designing Trading Strategies and Mechanisms for Electronic Markets (AMEC)*, 2010.

[12] P. Stone and S. Kraus. To teach or not to teach? decision making under uncertainty in ad hoc teams. In *Proc. of AAMAS'10*, 2010.

[13] P. Young. The evolution of conventions. *Econometrica*, 61(1):57–84, 1993.

Comparative Evaluation of MAL Algorithms in a Diverse Set of Ad Hoc Team Problems

Stefano V. Albrecht
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, UK
s.v.albrecht@sms.ed.ac.uk

Subramanian Ramamoorthy
School of Informatics
University of Edinburgh
Edinburgh EH8 9AB, UK
s.ramamoorthy@ed.ac.uk

ABSTRACT

This paper is concerned with evaluating different multiagent learning (MAL) algorithms in problems where individual agents may be heterogenous, in the sense of utilizing different learning strategies, without the opportunity for prior agreements or information regarding coordination. Such a situation arises in *ad hoc team* problems, a model of many practical multiagent systems applications. Prior work in multiagent learning has often been focussed on homogeneous groups of agents, meaning that all agents were identical and a priori aware of this fact. Also, those algorithms that are specifically designed for ad hoc team problems are typically evaluated in teams of agents with fixed behaviours, as opposed to agents which are adapting their behaviours. In this work, we empirically evaluate five MAL algorithms, representing major approaches to multiagent learning but originally developed with the homogeneous setting in mind, to understand their behaviour in a set of ad hoc team problems. All teams consist of agents which are continuously adapting their behaviours. The algorithms are evaluated with respect to a comprehensive characterisation of repeated matrix games, using performance criteria that include considerations such as attainment of equilibrium, social welfare and fairness. Our main conclusion is that there is no clear winner. However, the comparative evaluation also highlights the relative strengths of different algorithms with respect to the type of performance criteria, e.g., social welfare vs. attainment of equilibrium.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]

General Terms

Algorithms, Experimentation

Keywords

Multiagent Learning, Agent Coordination, Ad Hoc Teams

1. INTRODUCTION

Game theory provides a mathematically well defined framework for the analysis of multiagent interactive decision making problems. A game consists of a number of players, a set of actions for each player, and a payoff function for each player. A large portion of the theory is developed under the assumption that each player knows the structure of the game, i.e. the action sets and payoff functions of all players. This is known as a *complete information* game. A more complicated setting arises if the players have only partial information about the structure of the game, e.g. no player knows the payoff function of any other player. This is called an *incomplete information* game. While this type of game has been studied for over half a century (notably by Harsanyi [9–12]), the problem of incomplete information in multiagent learning (MAL) has received relatively lesser attention. In a version of this setting, called the *ad hoc team* problem [29], one seeks to design an autonomous agent which is able to collaborate efficiently with a previously unknown group of agents, in the absence of any prior coordination between the agent and its counterparts chosen in an ad hoc way.

This problem is motivated by numerous practical and important applications. As we increasingly employ autonomous agents in a growing number of areas, ranging from teams of robots in automated factories to internet trading agents, it is likely that these agents will have to collaborate with each other in nontrivial ways. As teams are augmented and continually modified over a long lifetime of operation, we would like agents to be able to learn how to collaborate efficiently with other agents, despite not knowing – a priori – who they are. An example of such interaction is given by Stone and Kraus in [31]. Another example comes from the domain of human-robot interaction in which a robot has to collaborate with a human, especially when the robot is not given any specific information about the characteristics of the human participant it must collaborate with.

The literature on the ad hoc team problem includes a number of proposals for models of interaction and learning. One approach is to try to learn to categorise other agents according to their behaviour or to use a set of role templates to constrain the interactions, e.g. [2, 7]. While these are useful ideas, it is worth noting that many realistic interactive decision making problems involve one further feature not captured in these models – multiple agents that are simultaneously trying to learn and adapt their behaviour. In such a setting, the environment is inherently nonstationary [37] and the interactive behaviour needs more careful treatment. Motivated by this general issue, this paper reports on an empirical study into the behaviour of multiagent learning algorithms in ad hoc teams.

There are two open issues with current MAL algorithms

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

in the literature. First, we note that most MAL algorithms (e.g. [1,3,5,6,8,14,17,23,32]) were primarily evaluated in homogeneous groups of agents, meaning that all agents in the group were identical, and all agents were a priori aware of this fact. Second, virtually all MAL algorithms which were specifically designed for ad hoc team problems (e.g. [2,7,30,31,36]) were primarily evaluated in teams of agents with fixed behaviours. In this work, we evaluate five MAL algorithms of the first sort in a series of ad hoc team problems. In addition, every team consists of agents which are continuously adapting their behaviours. The algorithms are evaluated in a comprehensive set of repeated games, ranging from games in which the players agree on what is most preferred, to games in which the players disagree on what is most preferred. Our performance criteria include the convergence rate, the final expected payoff, social welfare and fairness, and the rates of several solution types.

Our intention was to identify those approaches covered by our selection of algorithms which may be better suited for ad hoc team problems. However, as we will show, our results indicate that there is no clear favourite among the algorithms.

The remainder of this paper is structured as follows: Section 2 describes the experimental setup, including algorithms, games, and performance criteria. Section 3 presents and analyses the results of our experiments. Section 4 discusses related work and Section 5 concludes our work.

2. EXPERIMENTAL SETUP

2.1 Algorithms

Our selection of algorithms is motivated by the range of approaches it covers. We tested two algorithms that model their opponents and three that do not model their opponents:

- Joint Action Learning (JAL) [5,32]
- Conditional Joint Action Learning (CJAL) [1]
- Win or Learn Fast with PHC (WOLF-PHC) [3]
- Modified Regret-Matching[1] (RegMat) [14]
- Nash Q-Learning (NashQ) [17]

JAL tries to model its opponents by learning their marginal action probabilities. It uses these probabilities to compute the expected payoffs of all of its actions. CJAL extends JAL in that it learns the action probabilities of its opponents conditioned on its own actions. WOLF-PHC uses a hill climbing method in the space of mixed strategies to find an optimal strategy. RegMat minimises the regret it feels for not having played any other actions. Finally, NashQ tries to learn the payoff distributions of all agents and plays a Nash equilibrium strategy in each state, regardless of the actual behaviour of its opponents.

[1]We use the *Hannan-consistent* version of RegMat. This means that for each action $\hat{a}_i$, the Hannan regret $R_t(\hat{a}_i) = \frac{1}{t}\sum_{\tau=1}^{t} u_i(\hat{a}_i, a^{\tau}_{-i}) - \frac{1}{t}\sum_{\tau=1}^{t} u_i(a^{\tau})$ will be ≤ 0 as $t \to \infty$, where a^{τ} denotes the joint action played at time τ (see [14]).

Algorithm 1 Modified evaluation procedure

Initialise: Empty vector M_i for each agent $i \in N$
loop
  Randomly generate a strictly ordinal $2 \times 2 \times 2$ game Γ
  Randomly generate a team B from N with $|B| = 2$
  for all $i \in N$ **do**
    Play Γ with agents $\{i\} \cup B$, where agent i is player 1
    Compute metrics for agent i and store them in M_i
  end for
end loop
Return averaged metrics $avg(M_i)$ for each $i \in N$

2.2 Games

The algorithms are evaluated in a range of *repeated games*. A repeated game Γ is a tuple $(N, (A_i)_{i\in N}, (u_i)_{i\in N})$, where $N = \{1, ..., n\}$ is the set of players (or agents), A_i is the set of actions available to player i, and $u_i : A \to \mathbb{R}$ is the payoff function of player i, where $A = A_1 \times ... \times A_n$. In each repetition of the game, each player $i \in N$ simultaneously chooses an action $a_i \in A_i$ and receives the payoff $u_i(a_1, ..., a_n)$. Each player i chooses its actions based on a strategy $\pi_i : A_i \to [0, 1]$, which is a probability distribution over the set A_i. A strategy π_i is called a *pure* strategy if $\pi_i(a_i) = 1$ for some $a_i \in A_i$. A strategy is called a *mixed* strategy if it is not a pure strategy. Given a strategy profile $\pi = (\pi_1, ..., \pi_n)$, the expected payoff to player i is defined as $U_i(\pi) = \sum_{a_1,...,a_n} \pi_1(a_1) * ... * \pi_n(a_n) * u_i(a_1, ..., a_n)$.

Our experiments are divided into three parts. The first two parts evaluate the algorithms in the set of all structurally distinct strictly ordinal 2×2 no-conflict and conflict games, respectively (based on [25]). A $m_1 \times ... \times m_n$ game is one in which there are n players, each of which with m_i actions. In an *ordinal* game, each player ranks each of the $k = \prod_i m_i$ possible outcomes from 1 (least preferred) to k (most preferred). An ordinal game is called *strictly ordinal* if no two outcomes have the same rank. An ordinal game is called a *no-conflict* game if all players have the same set of most preferred outcomes, otherwise it is called a *conflict* game. Finally, a set of games is said to be structurally distinct of no game in the set can be reproduced by any transformation of any other game in the set. Possible transformations include interchanging the rows, columns, players, and any combination of these in the payoff matrix of the game.

We divided the games into no-conflict and conflict games because these define two distinct levels of difficulty. In a no-conflict game, it is relatively easy to arrive at a solution that is best for all players since all players have the same most preferred outcomes. However, in a conflict game, there is no such outcome. Therefore, the agents will have to arrange some form of a compromise. This requires reliable coordination mechanisms, especially in the context of ad hoc teams.

The third part of our experiments uses a modified version of the evaluation procedure proposed by Stone et al. [29]. The procedure tests the algorithms in a number of randomly generated strictly ordinal $2 \times 2 \times 2$ games. Each team may contain multiple agents of the same type. Every game is repeated for each algorithm under the same conditions. Algorithm 1 shows the pseudo-code of the procedure.

2.3 Performance criteria

This section provides definitions of the performance criteria

used in our experiments. The definitions are based on the notion of *plays* of repeated games. A play P_Γ of a repeated game Γ is a tuple $((\pi^t)_{t=1,...,t_f}, (a^t)_{t=1,...,t_f}, (r^t)_{t=1,...,t_f})$, where $t \in \{1, ..., t_f\}$ denotes the time, t_f denotes the final time of the play, $\pi^t = (\pi_1^t, ..., \pi_n^t)$ denotes the strategy profile at time t, $a^t = (a_1^t, ..., a_n^t)$ denotes the joint action at time t, and $r^t = (r_1^t, ..., r_n^t)$ denotes the joint payoff at time t.

2.3.1 Convergence rate

The convergence rate of an agent is defined as the percentage of plays in which the agent converged. Let P_Γ be a play of some repeated game Γ. We say that agent i converged in P_Γ if its strategies π_i^t stay within a tolerance bound of $\pm 5\%$ in the final 20% of the play. Formally, agent i converged in play P_Γ if for all $t \in \{t_s, ..., t_f\}$ and $j \in \{1, ..., m_i\}$, where $t_s = 0.8\, t_f$, we have $|\pi_i^{t_s}(j) - \pi_i^t(j)| \leq 0.05$.

A high convergence rate is not necessarily better than a low convergence rate. However, we argue that it may be useful if an agent has a high convergence rate as this means that the other agents will have to adapt to one less opponent (that is, once the agent has converged).

2.3.2 Final expected payoff

The final expected payoff of an agent is an approximation of the agent's expected payoff after having learned for t_f repetitions. The approximation is based on the final 20% of the play. Let P_Γ be a play of some repeated game Γ. The final expected payoff of agent i in play P_Γ is formally defined as $\bar{r}_i = \frac{1}{t_f - t_s + 1} \sum_{t=t_s}^{t_f} r_i^t$, where $t_s = 0.8\, t_f$. This is an important metric because it is a major indicator of the algorithm's individual performance.

2.3.3 Social welfare and fairness

Let P_Γ be a play of some repeated game Γ. We define the social welfare and fairness of play P_Γ, respectively, as the sum and product of the final expected payoffs $\bar{r}_i$ of all agents $i \in N$. Formally, this corresponds to $\sum_{i=1}^n \bar{r}_i$ for the social welfare, and $\prod_{i=1}^n \bar{r}_i$ for the social fairness. These metrics are useful for the assessment of an algorithm's team performance.

2.3.4 Rate of different solution types

We measure the rates of four different solution types. The definitions in the following sections rely on the notion of the *averaged final profile*. Let P_Γ be a play for some repeated game Γ. The averaged final profile (or AFP) of P_Γ is the average of all strategy profiles in the final 20% of the play. Formally, we denote the AFP by $\bar{\pi} = (\bar{\pi}_1, ..., \bar{\pi}_n)$, where $\bar{\pi}_i = \frac{1}{t_f - t_s + 1} \sum_{t=t_s}^{t_f} \pi_i^t$ and $t_s = 0.8\, t_f$. We use the final 20% of the play to approximate mixed strategies for agents that play pure strategies only (such as JAL and CJAL).

Nash equilibrium rate.

The Nash equilibrium (NE) rate is defined as the percentage of plays in which the AFP constitutes a Nash equilibrium. A strategy profile $\pi = (\pi_1, ..., \pi_n)$ is a Nash equilibrium if $U_i(\pi_1, ..., \pi_i, ..., \pi_n) \geq U_i(\pi_1, ..., \hat{\pi}_i, ..., \pi_n)$ for all players i and all strategies $\hat{\pi}_i$. Given a play P_Γ of a repeated game Γ, we determine if the play resulted in a Nash equilibrium by solving the following linear programme for each player $i \in N$:

$$\begin{aligned} \text{Maximise:} \quad & U_i(\bar{\pi}_1, ..., \pi_i, ..., \bar{\pi}_n) \\ \text{Subject to:} \quad & \forall j \in A_i : \pi_i(j) \geq 0 \\ & \textstyle\sum_{j \in A_i} \pi_i(j) = 1 \end{aligned}$$

We denote the optimised profile for player i by π^i. If, for any $i \in N$, the expected payoff under the optimised profile, $U_i(\pi^i)$, exceeds the expected payoff under the AFP, $U_i(\bar{\pi})$, by more than 5%, then we conclude that the play P_Γ did not result in a Nash equilibrium. Formally, we define P_Γ to result in a Nash equilibrium if $\forall i \in N : \frac{U_i(\pi^i)}{U_i(\bar{\pi})} \leq 1.05$.

Pareto optimality rate.

The Pareto optimality (PO) rate is defined as the percentage of plays in which the AFP is Pareto-optimal. A strategy profile π is Pareto-optimal if there is no other profile $\hat{\pi}$ such that $\forall i \in N : U_i(\hat{\pi}) \geq U_i(\pi)$ and $\exists i \in N : U_i(\hat{\pi}) > U_i(\pi)$. Similar to [1], we determine if a profile is Pareto-optimal by measuring its orthogonal distance to the *Pareto front*.

Consider a repeated game Γ. The space of possible expected (joint) payoffs of Γ is a convex polytope in $\mathbb{R}^n$ that has one dimension for each player $i \in N$. It is defined as the convex hull of all joint payoffs $(u_1(a), ..., u_n(a))$ for all joint actions $a \in A_1 \times ... \times A_n$. Each point in this payoff polytope corresponds to a tuple of expected payoffs $(U_1(\pi), ..., U_n(\pi))$ for some profile $\pi = (\pi_1, ..., \pi_n)$. The *Pareto front* of a payoff polytope Φ is defined as the set of Pareto-optimal faces of Φ. A face ϕ of Φ is Pareto-optimal if the corresponding strategy profiles of all points on ϕ are Pareto-optimal. Now, given a play P_Γ of Γ, we say that it results in a Pareto-optimal solution if the minimal orthogonal distance of its AFP, $\bar{\pi}$, when projected onto the payoff polytope Φ of Γ, to the Pareto front of Γ is not greater than 0.1. Tests indicate that this value works well for ordinal games.

Figure 1 shows the payoff polytope of a 2×2 game. The payoff table of the game is given in the figure. Player 1 chooses the row and player 2 chooses the column. The elements of the table contain the payoffs to player 1 and 2, respectively.

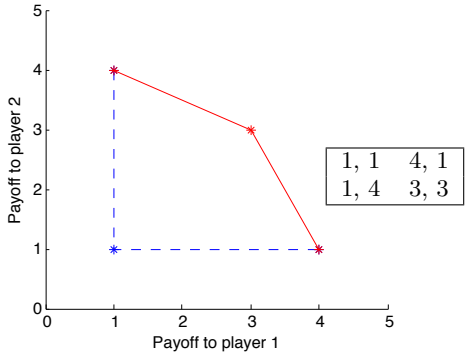


Figure 1: Example of a Pareto front.

The asterisks in the plot mark the four payoff pairs. The edges correspond to the faces of the polytope. Together, they define the convex hull of all payoff pairs. The solid edges show the Pareto-optimal faces of the polytope. Together, they define the Pareto front of the payoff polytope. This means that every strategy profile whose expected payoff is on[2] the Pareto front constitutes a Pareto-optimal profile.

[2]Or, as in our case, *close* to the Pareto front.

Agent	Conv.	Fexp.	Welfare	Fairness	NE	PO	WO	FO
JAL	1	3.9866	7.9720	15.9063	1	0.9920	0.9920	0.9920
CJAL	1	3.9831	7.9663	15.8874	1	0.9897	0.9897	0.9897
WOLF-PHC	0.9996	3.9449	7.8908	15.6426	1	0.9638	0.9638	0.9638
RegMat	0.9990	3.9107	7.8170	15.3906	0.9954	0.9457	0.9457	0.9457
NashQ	0.9987	3.9840	7.9733	15.9144	0.9954	0.9939	0.9939	0.9939

Table 1: Results for no-conflict games.

Agent	Conv.	Fexp.	Welfare	Fairness	NE	PO	WO	FO
JAL	0.8901	3.0140	6.0592	8.9997	0.8982	0.7781	0.7021	0.6164
CJAL	0.9456	3.0326	6.0978	9.0900	0.8470	0.8050	0.7184	0.6250
WOLF-PHC	0.9430	3.0392	6.0620	9.0517	0.9047	0.7636	0.6992	0.6142
RegMat	0.8673	3.0313	6.0368	8.9610	0.8946	0.7662	0.7000	0.6109
NashQ	0.9990	3.0446	6.0667	9.0755	0.8722	0.7767	0.6946	0.6097

Table 2: Results for conflict games.

Welfare/Fairness optimality rate.
The welfare optimality (WO) and fairness optimality (FO) rates are defined as the percentage of plays in which the AFP is welfare-optimal and fairness-optimal, respectively. Given a play P_Γ of a repeated game Γ, we say that it results in a welfare-optimal (fairness-optimal) solution if the welfare (fairness) of its AFP is not more than 5% lower than the maximum welfare (fairness) of Γ. The maximum welfare (fairness) of a game is the highest possible welfare (fairness) achievable by any strategy profile.

Given a game Γ, we compute its maximum welfare by solving the following non-linear optimisation problem:

$$\begin{array}{rl} \text{Maximise:} & \sum_{i\in N} U_i(\pi) \\ \text{Subject to:} & \forall i \in N\ \forall j \in A_i : \pi_i(j) \geq 0 \\ & \forall i \in N : \sum_{j\in A_i} \pi_i(j) = 1 \end{array}$$

Similarly, we compute its maximum fairness by solving the following non-linear optimisation problem:

$$\begin{array}{rl} \text{Maximise:} & \prod_{i\in N} U_i(\pi) \\ \text{Subject to:} & \forall i \in N\ \forall j \in A_i : \pi_i(j) \geq 0 \\ & \forall i \in N : \sum_{j\in A_i} \pi_i(j) = 1 \end{array}$$

We denote the optimised profile of the first problem by π^w and the one of the second problem by π^f. Then, for a given play P_Γ, we say that it resulted in a welfare-optimal solution if $\frac{W(\pi^w)}{W(\bar{\pi})} \leq 1.05$, where $W(\pi) = \sum_{i\in N} U_i(\pi)$. Similarly, we say that it resulted in a fairness-optimal solution if $\frac{F(\pi^w)}{F(\bar{\pi})} \leq 1.05$, where $F(\pi) = \prod_{i\in N} U_i(\pi)$.

2.4 Parameter settings and selection strategies

With the exception of RegMat, all algorithms are based on Q-learning [33]. This means that they use a table Q to store estimated values for each joint action $a \in A$. The values are updated using a formula of the form $Q(a) \leftarrow (1-\alpha)Q(a)+\alpha\, r$, where r is the payoff to the algorithm, and α is the learning rate. We use a constant learning rate of $\alpha = 0.1$ for all algorithms throughout all experiments. This violates the standard conditions for stochastic approximation [18], but enables the algorithms to learn continuously.

Furthermore, all algorithms except RegMat use a selection strategy to choose their actions. We use an ϵ-greedy strategy for all algorithms. Therein, the algorithm chooses a random action with probability ϵ, and the greedy action (i.e. the action that is currently believed to have the highest expected payoff) with probability $1-\epsilon$. We use a constant exploration rate of $\epsilon = 0.05$ for all algorithms. RegMat chooses its actions similar to ϵ-greedy. Here, the parameters δ and γ do the job (see [14]). We set these to $\delta = 0.1$ and $\gamma = 0.2$ for all experiments.

Finally, for WOLF-PHC, we need to specify two additional learning rates δ_w and δ_l with $\delta_l > \delta_w$ (see [3]). The algorithm uses the learning rate δ_w if it believes itself to be "winning", and it uses the rate δ_l if it believes itself to be "losing". We use a setting of $\delta_w = \frac{1}{1000+t}$ and $\delta_l = 2\delta_w$, where t denotes the time (or current repetition) of the game.

We tested these settings in a series of experiments and found them to be working well. Nonetheless, the settings are likely to be sub-optimal. This, however, is irrelevant for our purposes since we are not considering the rate at which the algorithms learn about the action values or explore the environment. This can be neglected insofar as that an optimised parameter setting, when considered in the long-run, will not lead to fundamental improvements (such as an enhanced capability to learn Nash equilibria). Instead, we chose to use identical or similar parameter settings for all algorithms in order to simplify the analysis of the results.

3. EXPERIMENTAL RESULTS

This section presents and analyses the results of our experiments. It is important to note that the performance of an algorithm may depend on the player position it takes on. To account for this, we repeated each play once for every permutation of the agent order. We call this process a *sweep*.

In the following, whenever we refer to statistical significance, this is based on a paired t-test with a significance level of 5%. We use the notation "Alg1 / Alg2" if the performances of the algorithms Alg1 and Alg2 are statistically equivalent (i.e. the difference is statistically insignificant). All reported results are averaged over all plays, games, and teams.

3.1 No-conflict games

The first part of our experiments evaluated the algorithms in the set of all structurally distinct strictly ordinal 2×2 no-conflict games (21 games in total). We evaluated all combinations of the algorithms in each of the games. In total, we evaluated each pair of algorithms in 25 sweeps (50 plays), where each play consisted of 100,000 repetitions.

Table 1 shows the performance metrics for every algorithm. The maximum payoff any player can achieve in any game is

Agent	Conv.	Fexp.	Welfare	Fairness	NE	PO	WO	FO
JAL	0.854	5.7964	17.2174	193.1478	0.804	0.712	0.466	0.396
CJAL	0.922	5.7856	17.3521	196.1594	0.760	0.742	0.486	0.418
WOLF-PHC	0.918	5.7400	17.1956	193.0255	0.824	0.676	0.442	0.388
RegMat	0.852	5.7290	17.2315	193.9011	0.844	0.708	0.438	0.392
NashQ	0.980	5.7562	17.2452	193.6470	0.790	0.734	0.452	0.388

Table 3: Results for random games.

4, and the maximum social welfare and fairness, respectively, are 8 and 16. All algorithms performed quite well. We note that NashQ, both in homogeneous and heterogeneous teams, performed extremely well. A closer look at the strategy trajectories reveals that NashQ persuades the other agent to play a NE strategy by playing a NE strategy itself, regardless of whether it could achieve a higher payoff by using another strategy. However, NashQ requires more information than any of the other algorithms.

Next, we note that those algorithms that model their opponents (i.e. JAL and CJAL) perform generally better than those that do not (i.e. WOLF-PHC and RegMat, leaving NashQ aside). Both JAL and CJAL have better results than WOLF-PHC and RegMat throughout all constellations of agents. Note also that JAL and CJAL have almost identical performances (all significance tests were negative). The results indicate that opponent modelling techniques may be better suited for ad hoc team scenarios because they learn the strategies of the other agents irrespective of the algorithms on which they are based. This is in line with Barrett et al. [2] and Wu et al. [36], whose algorithms are also based on opponent modelling techniques.

3.2 Conflict games

The second part of our experiments evaluated the algorithms in the set of all structurally distinct strictly ordinal 2×2 conflict games (57 games in total). As before, we evaluated each combination of the algorithms in each of the games, using 25 sweeps and 100,000 repetitions per play. The results are shown in Table 2. The maximum payoff any player can achieve in any of these games is 4. However, the maximum welfare and fairness vary among the games and can be as high as 7 and 12, respectively.

First, we note that NashQ achieved the highest convergence rate, followed by CJAL / WOLF-PHC, then JAL, and then RegMat. The high convergence rate of NashQ is explained by the fact that it plays a NE strategy in each state, regardless of whether another strategy would provide higher payoffs. Therefore, as soon as NashQ has learned the payoff structure of the game, it will always play the same strategy. The low convergence rate of RegMat can be explained by the fact that it constantly tries to maintain (or restore) the Hannan consistency, forcing it to frequently change its strategy.

The final expected payoffs and the average welfare and fairness of all algorithms are very similar. Indeed, all of these are statistically equivalent. This is an interesting result since it does not correspond to the solution rates of the algorithms. Here, one can see that WOLF-PHC / JAL / RegMat have the highest NE rates, followed by NashQ and CJAL. On the other hand, CJAL has the highest PO rate, followed by all other algorithms with statistically equivalent PO rates. The same applies to the WO and FO rates, where CJAL again achieved the highest rate, while the other algorithms achieved equivalent rates. Furthermore, note that NashQ has the second lowest NE rate although it plays a NE strategy in every state. Other than in the previous section, playing a NE strategy in every state does not necessarily seem to persuade the other agent to play a NE strategy as well.

The results seem to indicate that CJAL may be better suited for ad hoc team problems than the other algorithms. It achieved the highest PO, WO, and FO rates, and these are significantly higher than those of the other algorithms. Moreover, it has the highest average welfare and fairness (yet these are statistically equivalent to the other algorithms). Note also that, apart from NashQ, it achieved the highest convergence rate. A high convergence rate is useful since it allows the other agents to adapt to a stationary opponent (that is, once it has converged). This is especially useful in ad hoc team scenarios because the agents may not always be able to resort to coordination strategies. However, note that CJAL has the lowest NE rate of all algorithms. Thus, whether CJAL is better suited for ad hoc team problems will ultimately depend on the solution concept that is considered most appropriate for this domain.

3.3 Random games

In the third part of our experiments we investigated how well the algorithms scale to ad hoc teams with more than two agents. We used a modified version of the evaluation procedure proposed by Stone et al. (see Section 2.2). Specifically, we tested each of the algorithms in 500 randomly generated strictly ordinal $2 \times 2 \times 2$ games.

Table 3 shows the performance metrics for every algorithm. The maximum payoff any player can achieve in any game is 8, while the maximum welfare and fairness may vary among the games. If the game happens to be a no-conflict game, then these values amount to 24 and 512, respectively. However, if the game is a conflict game, then the maximum welfare and fairness may assume any value as high as 23 and 448, respectively. Of all games generated in our experiments, 2% were no-conflict games and 98% were conflict games.

First, we note that NashQ has the highest convergence rate. This is because once the algorithm has learned the game structure, it will always play the same NE strategy. The second and third highest convergence rates were achieved by CJAL / WOLF-PHC and JAL / RegMat, respectively. It is interesting to see that CJAL / WOLF-PHC and JAL / RegMat have similar convergence rates, since, in both groups respectively, the first algorithm plays pure strategies while the second algorithm plays mixed strategies. One would expect the former to have a significantly lower convergence rate than the latter because those algorithms that play pure strategies need to change their strategies periodically in order to approximate mixed strategies.

The final expected payoffs (FEPs) of all algorithms are statistically equivalent. Note that NashQ achieved the third highest FEP. This is interesting because NashQ plays a NE

strategy regardless of whether or not another strategy provides higher payoffs. Moreover, it is remarkable that JAL, CJAL, WOLF-PHC, and RegMat have equivalent FEPs despite the fact that the former two play pure strategies while the latter two play mixed strategies. This could indicate that the impact on the average payoffs due to the constellation of agents in a team may not be as strong as one might otherwise expect.

The average welfare and fairness confirm our observations of Section 3.2. CJAL achieved a higher welfare and fairness than any other algorithm. The welfare and fairness of the other algorithms are statistically equivalent. This corresponds to the WO and FO rates of the algorithms, where CJAL achieved significantly higher rates than all other algorithms. In fact, in almost half of all plays, CJAL managed to arrive at a welfare-optimal solution, of which a majority was fairness-optimal as well. As noted earlier, this may be a valuable property for ad hoc team problems. An ad hoc agent that is able to collaborate with an unknown group of agents such that the overall performance of the entire group is optimised (in terms of welfare and fairness) may be better than an agent that attempts to optimise its own payoff only. However, we note again that this essentially depends on the priorities of both the ad hoc agent and the entire group.

Finally, consider the different solution rates. The highest NE rate was achieved by RegMat, followed by WOLF-PHC, JAL / NashQ, and then CJAL. Note that the NE rate of CJAL is relatively low when compared to the other agents. However, this is opposed by the PO rates. Here, CJAL / NashQ achieved the highest rate, followed by JAL / RegMat and WOLF-PHC. It is interesting that CJAL and NashQ achieved equivalent PO rates, despite the fact that CJAL was specifically designed to learn PO solutions [3], whereas NashQ was specifically designed to learn NE solutions [17]. Note also that the PO rate of WOLF-PHC is quite low, especially in comparison to its relatively high NE rate.

3.4 Overall results

Figure 2 shows the results averaged over all three parts. Note that we cannot take the average of the payoffs since we consider ordinal games. However, for the final expected payoffs, we first normalised the values by dividing through the respective maximum, after which we took the average of all results. Thus, the final expected payoffs in Figure 2 are to be read as percentages where 0% means that the algorithm always achieved its least preferred outcome, and 100% means that it always achieved its most preferred outcome.

The results show that there is, in fact, no algorithm which achieved an overall better performance in the experiments, that is, *no algorithm is generally better*. The following summary underlines this:

- JAL has the second lowest convergence rate with about 91.47%. This comes from the fact that it changes its strategy periodically in order to approximate a mixed strategy. Furthermore, it has the third highest payoff rate with 82.49%. This rate is statistically equivalent to the highest rates. The NE rate of JAL is the third highest with about 90%. Finally, it has the third highest PO rate (82.74%), the second highest WO rate (72%), and the second highest FO rate (66.81%).

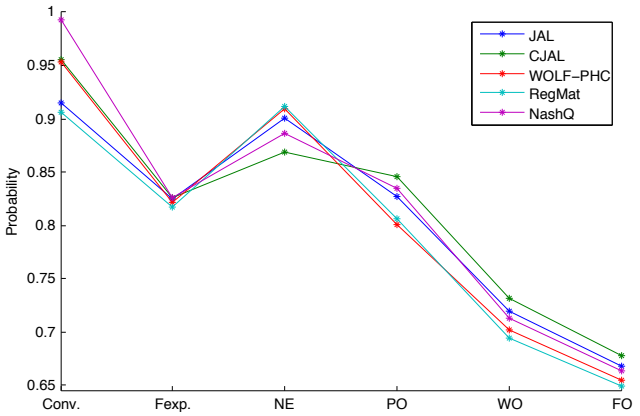


Figure 2: Overall results.

- CJAL has the second highest convergence rate with about 95.59%. This is since it attempts to learn Pareto-optimal solutions, which, in our case, are often pure profiles. Moreover, of all algorithms used in our experiments, it has the highest PO rate (84.56%), WO rate (73.14%), and FO rate (67.76%), which consequently leads to the highest payoff rate with about 82.57%. However, this is opposed by the fact that it achieved the lowest NE rate with 86.9%.

- WOLF-PHC has the third highest convergence rate with about 95.35%. This is statistically equivalent to CJAL. Furthermore, it has the second lowest payoff rate with 82.12%. On the other hand, with about 90.96% it has the second highest NE rate. This rate is statistically equivalent to the highest NE rate. Finally, it has the second lowest PO rate (80.12%), WO rate (70.17%), and FO rate (65.53%).

- RegMat achieved the lowest convergence rate with 90.61%. As pointed out earlier, it frequently changes its strategy in order to maintain the Hannan consistency. Moreover, with about 81.72%, it also has the lowest payoff rate. This is a consequence of its frequent changes. Interestingly, these efforts lead to the highest NE rate of all algorithms with about 91.14%. However, they also lead to the lowest PO rate (80.66%), WO rate (69.46%), and FO rate (64.95%).

- NashQ, with about 99.26%, managed to converge in most of the games. As was explained earlier, this is since it will always play the same strategy after it learned the payoff structure of the game. It is worth noting that it has the second highest payoff rate with about 82.56%, which is almost identical to the highest rate. This is interesting since NashQ chooses its strategies irrespective of the strategies of the other agents. On the other hand, it is surprising that it has the second lowest NE rate (88.59%), despite the fact that it plays a NE strategy in each state. It is equally surprising that it achieved the second highest PO rate (83.49%) and the third highest WO rate (71.35%) and FO rate (66.39%), despite the fact that it was not optimised for these solution types.

We conclude that the assessment of an algorithm ultimately depends on the solution concept that is considered most appropriate for the problem at hand. In other words, its performance depends on the priorities of the entire team. For example, in the predator domain investigated by Barrett et al. [2], it would be most desirable to arrive at a welfare-optimal solution if we define the welfare of the predator group to be the inverse of the average steps needed to capture the prey. Moreover, if we think of the agents as real robots (e.g. as in [31]), then we might want to arrive at a fairness-optimal solution in order to ensure that all robots have identical or similar energy consumptions. On the other hand, in a multiagent marketing application in which the other agents cannot be trusted (as they may want to deceive us in order to increase their payoffs), we would want to arrive at a Nash equilibrium (or minimax profile in 2-player zero-sum games) such that we can guarantee a minimum average payoff.

Although this conclusion may seem unsurprising at first glance, the implications should be of interest to researchers developing multiagent learning algorithms. The typical focus in this area has been on the concept of Nash equilibria (or, equivalently, minimax profiles in 2-player zero-sum games). For a selection, see [3–6,15–17,21–24,32]. Some authors focus on the concept of Pareto optimality as an alternative [1,20,27], others focus on correlated equilibria [8, 13, 14], and some do not make any specific commitments regarding the nature of the solution [2,30,31,36]. However, as can be seen from our results, it is important to consider a wider spectrum of solution concepts in order to fully assess the performance of an algorithm. Indeed, this bears an interesting resemblance to the No Free Lunch theorems of Wolpert and Macready [34, 35]. Therein, roughly speaking, it is argued that the performance of any two algorithms is identical when averaged over all possible problems. That is, whenever an algorithm is superior to another algorithm on a certain set of problems, this is paid for by inferiority on a different set of problems. Our results show a tradeoff relation of this kind. For instance, CJAL has the highest PO rate and the lowest NE rate whereas RegMat has the lowest PO rate and the highest NE rate. Other algorithms range somewhere in the middle, without best or worst performances. This seems to indicate that superiority in one solution type is compensated for by a converse relation somewhere else.

4. RELATED WORK

Harsanyi pioneered the study of incomplete information games. In his 1967 paper [10], he describes the *Bayesian game*, a game in which players have beliefs about missing information. He develops the concept of the Bayesian Nash equilibrium [11] in which each player plays a best response against the other players, based on the personal beliefs of the player. Jordan [19] showed that, for any repeated game, if the players play a Bayesian Nash equilibrium in each repetition, and if the personal beliefs of the players satisfy certain conditions, then this will converge to a true Nash equilibrium.

The problem of incomplete information in multiagent learning, in the form of the ad hoc team problem, was addressed by Stone et al. [29]. They propose a procedure to evaluate two ad hoc agents for a given set of potential team members and tasks. We used a modified version of this procedure for our own experiments (see Section 2.2).

In earlier work, Stone and Kraus [31] define optimal strategies for an ad hoc agent collaborating with a fixed-behaviour teammate in an environment modelled as a k-armed bandit. Stone et al. [30] present an algorithm that would lead a fixed greedy agent towards an optimal joint action in a simple repeated game in which both agents have identical payoff functions.

More recently, Genter et al. [7] introduced a framework for a *role-based* approach to the ad hoc team problem. Using a set of predefined roles (i.e. behaviours), the ad hoc agent tries to assume a role such that the marginal utility of the team is maximised. The agent was shown to be effective in several instances of the Pac-Man domain. However, we note that the framework is based on a number of key assumptions. It assumes that all teammates follow one of a finite set of a priori predefined roles, that the ad hoc agent knows what roles its teammates follow, and that the ad hoc agent knows the internal payoff distributions of its teammates.

These assumptions are relaxed in a recent empirical study by Barrett et al. [2]. They used an ad hoc agent that tries to identify its teammates by observing their behaviour and comparing it with a database of known behaviours. In addition, it learns a new model for the observed behaviour using a tree classifier. The agent combines both the database and the learned model in a Bayesian fashion to anticipate the behaviour of its teammates. Experiments showed that the ad hoc agent performed quite well, and in general better than those agents that just mimic their teammates.

Wu et al. [36] proposed an interesting algorithm called *Online Planning for Ad Hoc Agent Teams* (OPAT). For each encountered state, the algorithm estimates the values of all joint actions using Monte-Carlo Tree Search. These values are used to generate a stage game (i.e. a repeated game with one repetition), based on which the algorithm decides which action to take. The decision process considers the past m plays of the current stage game to approximate the strategies of the other agents. OPAT was shown to be effective in a series of multiagent domains.

5. CONCLUSION

In this work, we compared the performance of five multiagent learning algorithms in a set of ad hoc team problems. The algorithms were evaluated in a comprehensive range of repeated games, and the teams consisted of agents which were themselves learning. Our intention was to characterise the performance of salient types of multiagent learning algorithms in ad hoc team problems. Our experiments show that there is no clear favourite among the algorithms. In particular, we conclude that the performance of an algorithm ultimately depends on the solution concept that is considered most appropriate for the problem at hand.

The experiments in this paper were based on 2-player and 3-player matrix games, in order to make comparative statements in a well defined and comprehensive set of game types. It would be of interest to extend this analysis to the case of multi-player games (i.e. n-player games with $n > 3$), and to games with multiple states (i.e. stochastic games [28]). We expect that such extensions would bring many known difficulties regarding interactive decision making [37] to the fore and perhaps differentiate the multiagent learning algorithms further. We anticipate that going down this path may also clarify when one may need to draw on more elaborate models than stochastic games, e.g., as in [26].

6. REFERENCES

[1] D. Banerjee and S. Sen. Reaching pareto-optimality in prisoner's dilemma using conditional joint action learning. *Autonomous Agents and Multi-Agent Systems*, 15(1):91–108, 2007.

[2] S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *Proceedings of the 10th International Conference on Autonomous Agents and Multiagent Systems*, May 2011.

[3] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[4] G. Brown. Iterative solution of games by fictitious play. In T. Koopmans, editor, *Activity Analysis of Production and Allocation*. Wiley, 1951.

[5] 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.

[6] V. Conitzer and T. Sandholm. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *Proceedings of the 20th International Conference on Machine Learning*, volume 20, pages 83–90, 2003.

[7] K. Genter, N. Agmon, and P. Stone. Role-based ad hoc teamwork. In *Proceedings of the Plan, Activity, and Intent Recognition Workshop at the 25th Conference on Artificial Intelligence*, August 2011.

[8] A. Greenwald and K. Hall. Correlated q-learning. In *Proceedings of the 20th International Conference on Machine Learning*, pages 242–249, 2003.

[9] J. Harsanyi. Bargaining in ignorance of the opponents' utility function. *Journal of Conflict Resolution*, 6(1):29–38, 1962.

[10] J. Harsanyi. Games with incomplete information played by "bayesian" players, i-iii. part i. the basic model. *Management Science*, 14(3):159–182, 1967.

[11] J. Harsanyi. Games with incomplete information played by "bayesian" players, i-iii. part ii. bayesian equilibrium points. *Management Science*, 14(5):320–334, 1968.

[12] J. Harsanyi. Games with incomplete information played by "bayesian" players, i-iii. part iii. the basic probability distribution of the game. *Management Science*, 14(7):486–502, 1968.

[13] S. Hart and A. Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000.

[14] S. Hart and A. Mas-Colell. A reinforcement procedure leading to correlated equilibrium. *Economic Essays: A Festschrift for Werner Hildenbrand*, pages 181–200, 2001.

[15] J. Hu and M. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, volume 242, page 250, 1998.

[16] J. Hu and M. Wellman. Experimental results on q-learning for general-sum stochastic games. In *Proceedings of the 17th International Conference on Machine Learning*, page 414. Morgan Kaufmann Publishers Inc., 2000.

[17] J. Hu and M. Wellman. Nash q-learning for general-sum stochastic games. *The Journal of Machine Learning Research*, 4:1039–1069, 2003.

[18] T. Jaakkola, M. Jordan, and S. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, 1994.

[19] J. Jordan. Bayesian learning in normal form games. *Games and Economic Behavior*, 3(1):60–81, 1991.

[20] S. Kimbrough and M. Lu. Simple reinforcement learning agents: Pareto beats nash in an algorithmic game theory study. *Information Systems and E-Business Management*, 3(1):1–19, 2005.

[21] C. Lemke and J. Howson. Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2):413–423, 1964.

[22] M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, volume 157, page 163, 1994.

[23] M. Littman. Friend-or-foe q-learning in general-sum games. In *Proceedings of the 18th International Conference on Machine Learning*, ICML '01, pages 322–328. Morgan Kaufmann Publishers Inc., 2001.

[24] M. Littman and C. Szepesvári. A generalized reinforcement-learning model: Convergence and applications. In *In Proceedings of the 13th International Conference on Machine Learning*, pages 310–318. Morgan Kaufmann, 1996.

[25] A. Rapoport and M. Guyer. A taxonomy of 2×2 games. *General Systems: Yearbook of the Society for General Systems Research*, 11:203–214, 1966.

[26] T. Schelling. *The Strategy of Conflict*. Harvard University Press, 1980.

[27] S. Sen, S. Airiau, and R. Mukherjee. Towards a pareto-optimal solution in general-sum games. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 153–160. ACM, 2003.

[28] L. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39(10):1095, 1953.

[29] P. Stone, G. Kaminka, S. Kraus, and J. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *Proceedings of the 24th Conference on Artificial Intelligence*, July 2010.

[30] P. Stone, G. Kaminka, and J. Rosenschein. Leading a best-response teammate in an ad hoc team. In *Agent-Mediated Electronic Commerce: Designing Trading Strategies and Mechanisms for Electronic Markets*, pages 132–146, November 2010.

[31] P. Stone and S. Kraus. To teach or not to teach? decision making under uncertainty in ad hoc teams. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, May 2010.

[32] W. Uther and M. Veloso. Adversarial reinforcement learning. Technical report, Computer Science Department, Carnegie Mellon University, 1997.

[33] C. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[34] D. Wolpert and W. Macready. No free lunch theorems for search. Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

[35] D. Wolpert and W. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[36] F. Wu, S. Zilberstein, and X. Chen. Online planning for ad hoc autonomous agent teams. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, 2011.

[37] H. Young. *Strategic learning and its limits*. Oxford University Press, 2004.

An Analysis Framework for Ad Hoc Teamwork Tasks

Samuel Barrett and Peter Stone
Dept. of Computer Science
The University of Texas at Austin
Austin, TX 78712 USA
{sbarrett, pstone}@cs.utexas.edu

ABSTRACT

In multiagent team settings, the agents are often given a protocol for coordinating their actions. When such a protocol is not available, agents must engage in ad hoc teamwork to effectively cooperate with one another. A fully general ad hoc team agent needs to be capable of collaborating with a wide range of potential teammates on a varying set of joint tasks. This paper presents a framework for analyzing ad hoc team problems that sheds light on the current state of research and suggests avenues for future research. In addition, this paper shows how previous theoretical results can aid ad hoc agents in a set of testbed domains.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Ad Hoc Teams, Multiagent Systems, Teamwork

1. INTRODUCTION

As the number of autonomous agents in society grows, so does the need for them to interact with other agents effectively. Both robots and software agents are becoming more common, and they are becoming more durable and robust, remaining deployed for increasing durations. Most existing methods for handling the interactions of agents require prior coordination, in the form of protocols for either coordination or communication. However, as agents stay deployed for longer, new agents are likely to be introduced that may not share these protocols. Furthermore, a multitude of different agents are under development in different businesses and research laboratories. Unfortunately, it is unlikely that these agents will all share a common world view or communication protocol. Therefore, it is desirable for agents to be capable of adapting to new teammates and learning to cooperate with previously unseen agents as part of an *ad hoc* team.

For example, consider a disaster rescue scenario where many different robots developed by many different people converge on the location of a disaster to attempt to locate and rescue victims. It is desirable for the robots to cooperate as a team, but in such a scenario, there is no time to program the robots to cooperate on site. If the robots were not explicitly designed to cooperate with one another, they will not work together and may even hinder each other. On the other hand, if some of these robots are programmed to reason about ad hoc teamwork, they may be able to quickly adapt their behaviors to cooperate with the other robots to rescue victims.

In a recent AAAI challenge paper, Stone et al. [15] introduced the concept of an *ad hoc team setting*, specifying it as a problem in which team coordination strategies cannot be specified a priori. They further presented a framework for evaluating the performance of an ad hoc team agent with respect to a domain and a set of possible teammates. The authors argued that the ad hoc teamwork challenge is inherently an empirical problem, but noted that little empirical research has been done in this area so far. On the other hand, some recent papers provide interesting theoretical results for some specific, isolated ad hoc team scenarios [16, 17].

This paper introduces a framework for understanding the relationships among existing lines of research, specifying several dimensions that are especially relevant when reasoning about the difficulty of different ad hoc team problems. Furthermore, this paper investigates several empirical scenarios and shows how existing theoretical solutions can be applied to these problems.

The remainder of the paper is organized as follows. Section 2 gives a more complete definition of ad hoc teamwork and specifies the empirical evaluation framework used in this work, and then Section 3 introduces a classification framework for ad hoc team problems along three important dimensions. Section 4 studies four different variations of an experimental domain, each representing a different point within the classification framework, and identifies their different solutions. Next, Section 5 explores other ad hoc teamwork domains, classifies them with respect to these dimensions, and suggests avenues for future research. Section 6 situates our research in literature, and Section 7 concludes.

2. AD HOC TEAMS

In an ad hoc team, agents need to cooperate with previously unseen teammates. Rather than developing protocols for coordinating an entire team, ad hoc team research focuses on developing agents that cooperate with teammates in the absence of such explicit protocols. Therefore, we consider a single agent cooperating with teammates that may or may not adapt to its behavior. We assume that we can only develop algorithms for the ad hoc team agent, without having any direct control over the other teammates.

For this work, we adopt essentially the same evaluation framework proposed by Stone et al. [15]. This framework is specified in Algorithm 1. According to this framework, the performance of the ad hoc team agent a depends on the distribution of problem domains D and the distribution of possible teammates A that it will

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

cooperate with. For the team B cooperating to execute the task d, $s(B, d)$ is a scalar score representing their effectiveness, where higher scores indicate better performance. The algorithm takes a sampling approach to average the agent's performance across a range of possible tasks and teammates to capture the idea that a good ad hoc team player ought to be robust to a wide variety of teamwork scenarios. We use s_{min} as a minimum acceptable reward for the team to be evaluated, because the ad hoc team agent may be unable to accomplish a task if its teammates are too ineffective, regardless of its own abilities. It is mainly used to reduce the number of samples required to evaluate the ad hoc agents and reduces the noise in the comparisons. Metrics other than the sum of the rewards can be used depending on the domain, such as the worst-case performance.

Algorithm 1 Ad hoc agent evaluation

Evaluate(a, A, D):

- Initialize performance (reward) counter $r = 0$.
- Repeat:
  - Sample a task d from D.
  - Randomly draw a subset of agents B, from A such that $E[s(B, d)] \geq s_{min}$.
  - Randomly select one agent $b \in B$ to remove from the team to create the team B^-.
  - Increment r by $s(\{a\} \cup B^-, d)$
- If Evaluate(a_0, A, D) > Evaluate(a_1, A, D) and the difference is significant,then we conclude that a_0 is a better ad hoc team player than a_1 in domain D over the set of possible teammates A.

3. DIMENSIONS OF AD HOC TEAM PROBLEMS

Section 2 specified the framework for evaluating ad hoc team agents, but this evaluation depends on the specific domain and teammates that the ad hoc agent may encounter. In this section, we identify three dimensions of ad hoc teamwork settings that we believe can be used to better understand these domains and teammates. There are many possible ways that ad hoc team domains can vary, such as the size of the task's state space and the stochasticity of the domain. But we find that for differentiating among the algorithms in the existing literature, the following three are most informative.

1. **Team Knowledge:** Does the ad hoc agent know what its teammates' actions will be for a given state, before interacting with them?
2. **Environment Knowledge:** Does the ad hoc agent know the transition and reward distribution given a joint action and state before interacting with the environment?
3. **Reactivity of teammates:** How much does the ad hoc agent's actions affect those of its teammates?

These dimensions affect the difficulty of planning in the domain in addition to how much an ad hoc agent needs to explore the environment and its teammates. When an ad hoc agent has good knowledge, it can plan without considering exploration, but when it has incomplete knowledge, it must reason about the cost and benefits of exploration. The exploration-exploitation problem has been studied previously, but adding in the need to explore the teammates' behaviors and the ability to affect them considerably alters this tradeoff. Sections 3.1–3.3 provide further details about each of these dimensions, how we measure them, and why they are important for ad hoc teamwork.

To better illustrate the dimensions, we introduce a simple domain to evaluate across each of the dimensions. We describe the domain here and revisit it in the discussion of each dimension.

MatchActions: *This domain is a typical coordination game with two agents, each of which has two actions. If they select the same action, both receive a reward of r_i, where r_i is randomly selected from $\{0.5, 0.75, 1.0\}$ for $i \in 1, 2$, but fixed for the episode. On the other hand, if both agents select different actions, they receive a reward of 0. In addition, both agents can observe their teammates' previous actions. The ad hoc agent knows that its teammate is following one of two behaviors:*

- ***FirstAction**: the teammate always chooses the first action*
- ***BestResponse**: the teammate chooses the same action as the ad hoc agent did previously*

Therefore, the state can be represented as the previous action taken by the ad hoc agent, called s_0 if the ad hoc agent chose the first action, and s_1 otherwise.

3.1 Team Knowledge

The ad hoc agent's knowledge about its teammates' behaviors gives insight into the difficulty of planning in the domain. The agent's knowledge can range from knowing the complete behaviors of its teammates to knowing nothing about them. Settings with partial information are especially relevant, because in many real world problems, the exact behavior of a teammate may not be known, but some reasonable guidelines of their behaviors exist. For example, when playing soccer, one can usually assume that a teammate will not intentionally pass to the other team or shoot at the wrong goal. If the behaviors are completely known, the agent can reason fully about the team's actions, while if the behaviors are unknown, the agent must learn about them and adapt to find a good behavior.

To estimate the ad hoc agent's knowledge about its teammates' behaviors, we compare the actions the ad hoc agent expects them to take and the ground truth of what actions they take. Specifically, we compare the expected distribution of teammate actions to the true distribution that the teammates follow. To compute the difference between the distributions, we use the Jensen-Shannon divergence measure, which was chosen because it is a smoothed, symmetric variant of the popular Kullback-Leibler divergence measure. When the ad hoc agent has no information about a teammate's action, we assume that it uses the uniform distribution to represent its actions. Therefore, we define the knowledge measure as

$$K(T,P) = \begin{cases} 1 & \text{if } \mathrm{JS}(T,P) = 0 \\ 1 - \dfrac{\mathrm{JS}(T,P)}{\mathrm{JS}(T,U)} & \text{if } \mathrm{JS}(T,P) < \mathrm{JS}(T,U) \\ -\dfrac{\mathrm{JS}(P,U)}{\mathrm{JS}(U,\mathrm{Point})} & \text{otherwise} \end{cases} \quad (1)$$

where T is the true distribution, P is the predicted distribution, U is the uniform distribution, Point is a distribution with all weight on one point (e.g. $[1, 0, 0, \ldots]$), and JS is the Jensen-Shannon divergence measure. By this definition, $K(T, T) = 1$, so the knowledge is complete if the ad hoc agent knows the true distribution. $K(T, U) = 0$, representing when the ad hoc agent has no knowledge and relies on the uniform distribution. Finally, if the predicted distribution is less accurate than the uniform distribution, then $K(T, P)$ is negative, with a minimum value of -1. This measure captures the range [0,1] smoothly, but can still be used for the range [-1,0] [1]. However, we generally expect the prediction to

[1]One slight anomaly of this measure is that when T is the uniform distribution (e.g. [.5,.5]), K is either 1 when P is exactly correct at [.5 .5] or negative. For all other values of T, K smoothly spans the range [-1,1].

be a higher entropy distribution than the true distribution as the ad hoc agent ought to correctly model its uncertainty in its teammates' behaviors rather than being confident and wrong, which keeps the measure in the range [0,1].

We define the ad hoc agent's knowledge about its teammates' behaviors as

$$\text{TeamK} = \frac{\sum_{s=1}^{n}\sum_{t=1}^{k} K(\text{TrueAction}_t(s), \text{ExpAction}_t(s))}{nk}$$

where $1 \leq s \leq n$ is the state, $1 \leq t \leq k$ specifies a teammate, $\text{TrueAction}_t(s)$ is the ground truth action distribution for teammate t for state s, and $\text{ExpAction}_t(s)$ is the action distribution that the ad hoc agent expects teammate t to select for state s.

We assume that $\text{ExpAction}_t(s)$ is the uniform distribution if the agent has no information about teammate t's actions in state s. Thus, if the ad hoc agent has better information about its teammates' behaviors, the distance between the distributions will be smaller and TeamK will be higher.

Let us now calculate the TeamK for the MatchActions domain. The ad hoc agent has uniform beliefs over its teammate following either the FirstAction or BestResponse behaviors. However, the teammate is actually following the BestResponse behavior. With these beliefs, in s_0, the ad hoc agent expects that its teammate will always chose a_0, so $\text{ExpAction}_{s_0} = [1, 0]$. In s_1, the ad hoc agent thinks that the teammate will choose a_0 with probability 0.5 and a_1 with probability 0.5, while it actually chooses a_1 with probability 1. Thus,

$$\text{TeamK} = \frac{K([1,0],[1,0]) + K([0,1],[\frac{1}{2},\frac{1}{2}])}{2} = \frac{0+1}{2} = 0.5$$

This indicates that the ad hoc agent is fairly knowledgeable about its teammate's actions.

3.2 Environmental Knowledge

Another informative dimension is how much knowledge the ad hoc agent has about the effects of a joint action given a state, for example the transition and reward functions. If the ad hoc agent has complete knowledge about the environment, it can plan about what actions it should select more simply than if it must also consider unknown effects of actions. However, if it has incomplete knowledge, it must explore its actions and face the standard problem of balancing exploring the environment versus exploiting its current knowledge.

Similarly to teammate knowledge, we formally define the ad hoc agent's knowledge about the environment's transitions as

$$\text{TransK} = \frac{1}{nm}\sum_{s=1}^{n}\sum_{j=1}^{m} K(\text{TrueTrans}(s,j), \text{ExpTrans}(s,j))$$

where $1 \leq s \leq n$ is the state, $1 \leq j \leq m$ is a joint action, K is taken from Equation (1), $\text{TrueTrans}(s,j)$ is the ground truth transition distribution from state s given joint action j, and ExpTrans is the ad hoc agent's expected transition distribution. If the agent has no information about the transitions, we assume that $\text{ExpTrans}(s,j)$ is the uniform distribution. Intuitively, if the ad hoc agent knows more about the transition function, then the distance between TrueTrans and ExpTrans will be smaller and as a result TransK will be higher. We define the agent's knowledge about the environmental rewards similarly, and let $\text{EnvK} = (\text{TransK}, \text{RewardK})$.

Revisiting the MatchActions domain, the ad hoc agent knows the true transition function, as it only depends on the ad hoc agent's previous action, so $\text{TransK} = 1$. However, it only knows that the payoff for each action is uniformly drawn from {0.5, 0.75, 1.0} and the reward is 0 if the agents' actions do not match. There are 8 possible cases to count over, coming from 2 states and 2 actions for each of the agents, but the cases fall into 2 sets based on whether the actions match. In addition, it does not matter which value each matched action actually takes, so we can simplify the calculation. Note that there are four reward values possible: {0, 0.5, 0.75, 1.0}. This leads to

$$\text{RewardK} = \frac{4*1 + 4*K([0,1,0,0],[0,\frac{1}{3},\frac{1}{3},\frac{1}{3}])}{8} = 0.582$$

Thus, $\text{EnvK} = (1, 0.582)$ As the agent observes these payoffs, it can refine its knowledge, but we are evaluating these properties prior to the ad hoc agent interacting with its environment.

3.3 Teammate Reactivity

The optimal behavior for the ad hoc agent also depends on how much its teammates react to its actions. If its teammates' actions do not depend on the ad hoc agent at all, the ad hoc agent can simply choose its actions to maximize the team reward, as if it were a single agent problem. Considering the actions of its teammates separately from that of the environment may still help computation by factoring the domain. However, if the teammates' actions depend strongly on the ad hoc agent's actions, the ad hoc agent's reasoning should consider what its teammates' reactions will be. If the ad hoc agent is modeling its teammates and its teammates are modeling the ad hoc agent, the problem can become recursive, as is directly addressed by Vidal and Durfee's Recursive Modeling Method [21].

A formal measure of the teammate reactivity needs to capture how different the teammates' actions will be when the ad hoc agent chooses different actions. We measure the distance between the resulting distributions of the teammate joint actions, using the pairwise Jensen-Shannon divergence measures. However, it is desirable for the distance to be 1 when the distributions have no overlap, so we use a normalizing constant of $\log 2$. Thus, we define the *reactivity* of a domain in state s as

$$\text{Reactivity}(s) = \frac{1}{(m-1)^2 \log 2}\sum_{a=1}^{m}\sum_{a'=1}^{m} \text{JS}(T(s,a), T(s,a'))$$

where JS is the Jensen-Shannon divergence measure, $1 \leq a, a' \leq m$ is the actions available to the ad hoc agent, and $T(s,a)$ is the distribution of the teammates' joint actions given the state s and the ad hoc agent's action, a. We use $m-1$ in the denominator because we exclude the case where $a = a'$; in the numerator, the JS measure will be 0 in this case. For the overall reactivity of the domain, we average over the states, resulting in $\text{Reactivity} = \frac{1}{n}\sum_{s=1}^{n} \text{Reactivity}(s)$. It is possible to consider how an action affects the teammates' actions further in the future, but we restrict our focus to one step reactivity for this paper. Note that all of the sums in this formulation can be converted to integrals for continuous states or actions. This formulation is similar to the empowerment measure used by Jung et al. [13], but we consider the ad hoc agent's ability to change the actions of its teammates rather than the environment state.

Let us once again explore this dimension in the context of the MatchActions domain. Although the ad hoc agent is unsure of its teammate's behavior, the teammate is truly playing the BestResponse behavior. Thus, its actions are entirely dependent on the ad hoc agent's actions, so $\text{Reactivity} = 1$. If instead the teammate played BestResponse with probability $\frac{9}{10}$ and FirstAction with probability $\frac{1}{10}$, then we would get

$$\text{Reactivity}(s) = \frac{\text{JS}([1,0],[\frac{1}{10},\frac{9}{10}]) + \text{JS}([\frac{1}{10},\frac{9}{10}],[1,0])}{2\log 2} = 0.758$$

Therefore, we can conclude that the agent would still be very reactive, though not as reactive as the BestResponse agent.

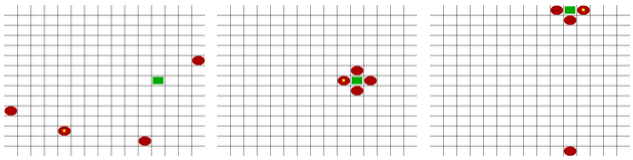
4. AD HOC TEAMWORK IN THE PURSUIT DOMAIN

Section 2 specified the framework for evaluating ad hoc team agents, but this evaluation depends on the specific domain and teammates that the ad hoc agent may encounter. Therefore, in this section, we study several concrete versions of a domain that require the cooperation of a team. Then, we explore how ad hoc agents should handle these various domains, and explain how these domains are characterized by the dimensions presented in Section 3.

4.1 The Pursuit Domain

The pursuit domain has become a popular setting for multiagent research [18]. It lends itself well to ad hoc team problems as it requires multiple agents to cooperate to capture the prey. The general idea of the pursuit problem is that a number of predators attempt to chase and finally "capture" a prey, but there are several variations of the pursuit, depending on the number of predators, the definition of "capture," and the mechanics of the world. Here we specify a number of variations of the pursuit domain that are interesting for investigating ad hoc teamwork.

4-Predator Known Behaviors Pursuit (4PKB): *In this fairly common formulation of the pursuit problem, there are four predators trying to capture a single prey, while moving around a toroidal grid, where moving off one side brings the agent back on the other side. Therefore, all four predators are required to capture the prey by surrounding it, as illustrated in Figure 1. In this formulation, all agents can fully observe the positions of the other agents and the prey moves randomly. One of the predators is an ad hoc agent and, at each time step, it must choose a direction to move to cooperate with its teammates. The other three predators follow a fixed behavior that is known to the ad hoc agent.*



(a) Random start (b) Capture position (c) Another capture

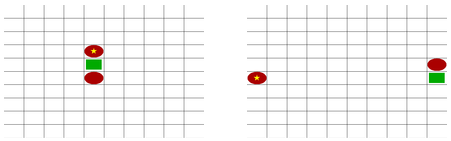
Figure 1: Start and capture positions in the 4PKB and 4PUB domains. The green rectangle is the prey, the red ovals are predators, and the red oval with the star is the ad hoc predator (the one under our control that is being evaluated).

Before continuing with describing the other domain variants, we introduce a number of high level behaviors that the predators may use to capture the prey. Specifically, we consider the following four individual predator behaviors described in Barrett et al.'s work [1]:

1. Greedy (**GR**)- Move towards the nearest unoccupied cell neighboring the prey with minimal obstacle avoidance
2. Greedy Probabilistic (**GP**) - Same as the GR behavior except that the predator has a chance of taking a longer path to its desired cell
3. Teammate-aware (**TA**) - Assign cells neighboring the prey to the teammates, minimizing the movement required by the farthest predator; move towards the assigned cell
4. Probabilistic Destinations (**PD**) - Spread out from other predators into a circle that tightens around the prey over time

4-Predator Unknown Behaviors Pursuit (4PUB): *This version is identical to 4PKB, except that the ad hoc agent is not given full information about its teammates' behaviors. Instead, the agent is given a set of known behaviors that its teammates are possibly playing. In this case, the ad hoc agent must observe its teammates and try to determine their behaviors based on their actions. For example, the ad hoc agent may be initially given a uniform distribution over the GR, GP, TA, and PD behaviors. By observing its teammates' actions, the ad hoc agent may be able to determine that all of its teammates are following the TA behavior.*

2-Predator Simultaneous Pursuit (2PS): *In this formulation of the pursuit problem, two predators move on a toroidal grid and attempt to capture the prey by simultaneously occupying any two cells neighboring the prey, as shown in Figure 2. Instead of choosing an action at every time step of an episode, the agents choose a high level behavior to play for the duration of an episode. This high level behavior defines their actions at each time step. In between episodes, the agents can choose a new behavior to play, based on their previous experience. Once again, one of the predators is controlled by the ad hoc agent, and its teammate chooses the behavior by best response, using a memory bound of k. For example, at the beginning of each episode, each predator could choose to play the GR, GP, TA, or PD behavior. If $k = 1$ and the ad hoc agent chose the GR behavior last step, its teammate would choose to play the GR behavior this step because it knows that this will result in the shortest time to capture the prey if the ad hoc agent continues playing the GR behavior.*



(a) Capture position (b) Another capture

Figure 2: Capture positions in the 2PS domain

2-Predator Teaching Pursuit (2PT): *In this version, two predators are trying to capture the prey by choosing a high level behavior to follow, similar to the 2PS domain. However, each predator must capture the prey independently, and the predators alternate episodes. Therefore, instead of cooperating during an episode, the predators cooperate between episodes. Each predator observes what high level behavior the other chose as well as how long it takes to capture the prey using that behavior. In addition, the predators are still a team and share rewards. In this domain, capture is defined as the predator occupying the same cell as the prey. For example, if the ad hoc agent chooses to play the GR behavior, its teammate observes that it chose the GR behavior as well as the length of the episode. The teammate can then use this information when it is selecting a behavior to play for the next episode.*

In addition, the ad hoc team agent has full knowledge about the performance of the behaviors, while its teammate starts with no knowledge and acts greedily with respect to the behaviors' observed sample means. If the ad hoc agent was not on a team, it could perform optimally by choosing the behavior with the best expected reward, but its teammate can observe its actions and learn from them. The ad hoc agent knows that its teammate is greedy with respect to the observed means of the different behaviors. However, the teammate has noisy actuation, so it is unable to perform deterministic behaviors, unlike the ad hoc agent. Unfortunately, the behavior with the best expected time to capture the prey is deterministic. Therefore, there is a cost to teaching as the ad hoc team agent must forego playing the best behavior to increase its teammates knowledge.

4.2 Prior Ad Hoc Teamwork Results

In Section 4.1, several variations of the pursuit domain were presented, some of which have been studied in prior research. Both the 4PKB and 4PUB domains were investigated by Barrett et al. [1]. In

their work, Barrett et al. assume that three of the predators use one of the specified behaviors (the same one in most cases) and the fourth predator is the ad hoc team agent. Their ad hoc team agent plans efficiently using Monte Carlo Tree Search (MCTS), selecting actions that are expected to capture the prey quickly given its models of its teammates. In the 4PKB domain, the ad hoc agent knows the true behavior of its teammates, but in the 4PUB domain, it is only given a set of possible behaviors of its teammates. Therefore, the ad hoc agent agents tracks the probabilities that its teammates are using the known behaviors, updating their probabilities using Bayes' rule and the probability of each behavior taking the observed actions. At each time step, the agent plans using MCTS and samples from the possible teammate models with respect to their relative probabilities. This approach results in an effective ad hoc team agent.

In Barrett et al.'s work, the ad hoc agent has complete information about the environment (EnvK $= (1, 1)$), but its knowledge about its teammates is varied in different tests. In the 4PKB setting, the ad hoc agent knows the true behavior of its teammates, so TeamK $= 1$. In the 4PUB setting, it only knows that its teammates are drawn from a set of known models; resulting in TeamK $= 0.720$ for a 5x5 world, while on a 20x20 world TeamK $= 0.807$. Finally, there are tests where the set of representative behaviors are known to the agent, but the teammates' behaviors are not drawn from this set. Instead, these agents are sampled from a set of predator behaviors written by students for a class assignment. In this case, TeamK $= 0.155$ on a 5x5 world and TeamK $= 0.237$ on a 20x20 world.

In this work, the reactivity of the teammates depends on the behavior that the teammates run as well as the size of the world. For the GP teammates on a 5x5 grid, the reactivity is only 0.0635, while if the teammates play the TA behavior, the reactivity is 0.501. Similarly, on a 20x20 grid, the reactivity of GP teammates is 0.00105 and for TA teammates it is 0.0809.

In the pursuit domain, the challenge arises from a combination of the reactivity of the teammates and the agent's imperfect knowledge about its teammates. In this research, only a single episode was considered, so there was no long term learning; the agents had to learn during the episode.

While Barrett et al. investigated the 4PKB and 4PUB domains, the pursuit domain allows for many small variations that have a large impact on where it falls along the dimensions laid out in Section 3. Sections 4.3–4.4 explore the 2PS and 2PT variants, all within the context of our framework from Section 3.

4.3 Repeated Interactions with a Best Response Agent

Whereas both the 4PKB and 4PUB domains assume that the ad hoc agent interacts with its teammates for only one episode, many teamwork settings allow for multiple interactions among the same teammates. In this case, long-term learning (across episodes) is both possible and very useful. In order to model such settings, in this section we investigate the 2PS domain.

In the 2PS domain, the ad hoc agent has perfect information about its teammate, so TeamK $= 1$. Also, the ad hoc agent completely knows the environment's transitions and rewards, so EnvK $= (1, 1)$. The reactivity of the domain is high, since the teammate's actions depend highly on the ad hoc agent's actions. However, the reactivity depends on the specific behaviors that are available to the agents as well as the memory size of the teammate, k. The available information about the teammate reduces the difficulty of this problem compared to the earlier pursuit problem, but the reactivity of this problem is much higher. Therefore, the problem is still difficult, but many of the issues that must be faced are different.

Analysis of 2PS reveals that it can be modeled as a repeated normal-form game in which the agents share the payoffs. In this setting, there is a matrix of shared payoffs and two agents; one chooses a row and one that chooses a column, where the rows and columns correspond to different behaviors that can be chosen. One of the agents is a k-memory bounded, best response agent, meaning that it chooses the action that has the best expected payoff given the other agent's last k actions. The other agent is the ad hoc team agent, and its goal is to cooperate with the best response agent to achieve the highest payoff.

There is a cell in the payoff matrix with the highest reward that is best for both agents. However, if the ad hoc agent jumps immediately to the corresponding behavior, it may incur a high loss before the best response agent moves to the best action, where the loss is defined as the difference between the maximum possible reward and the received reward. Therefore, it may be desirable for the ad hoc agent to take a longer path through the payoffs, minimizing the losses.

Stone et al. [16] investigated the class of ad hoc teamwork problems that can be modeled by this normal-form game formulation. Their work provides several theoretical results as well as an efficient algorithm for finding the optimal action sequence when $k = 1$. Furthermore, they give an algorithm for dealing with larger memory bounds, $k > 1$, but this algorithm is exponential in the memory size. Also, they consider the case where the teammate is non-deterministic and differs from the k memory bounded best response by ϵ.

In this paper, the predators select from the TA, PD, and GR behaviors. If they play on a 5x5 grid and each agent has a 0.1 chance of taking a random action, the resulting payoff matrix is given in Table 1. These payoffs were calculated by running 1,000 episodes with the agents following the specified behaviors, where the team receives an action penalty of -1 for each step until the prey is captured.

	TA	PD	GR
TA	-4.583	-5.123	-5.152
PD	-5.123	-4.946	-4.615
GR	-5.152	-4.615	-4.379

Table 1: Payoff matrix from the pursuit domain

Assume that the agents start with both agents playing the TA behavior (a Nash equilibrium) and $k = 1$. The best payoff is when both agents play the GR policy, so the ad hoc team agent wants to find the lowest cost path to that policy combination. This occurs if the ad hoc agent chooses the PD policy and then the GR policy from then on. The best response teammate will play the TA policy for the first two steps (because it is the best response to TA) then change to the GR policy, which is the best response to PD. The loss of this path is $-4.379 - -5.123 = 0.744$ while changing directly has a cost of $-4.379 - -5.152 = 0.773$. Therefore, it is advantageous for the ad hoc agent to take a longer path to its desired policy. The efficient algorithm laid out by Stone et al. finds this solution.

In this domain, both agents know the transitions and rewards, so EnvK $= (1, 1)$. Also, the ad hoc agent knows that its teammate uses the best response policy, resulting in TeamK $= 1$. The reactivity is 0.198, as the ad hoc agent's actions do influence its teammate's actions.

The version of the pursuit domain considered in this section illustrated how prior theoretical results can be applied directly in an experimental setting. In the next section, we see the same for a different theoretical approach to ad hoc teamwork.

4.4 Teaching a Novice Agent

To this point, we have assumed that the teammate has complete knowledge about the performance of the behaviors, but in some settings this is not the case. To explore such settings, we investigate the 2PT domain, in which the teammate starts with no knowledge about each behavior and must explore the behaviors to estimate their performance.

However, we assume that the ad hoc team agent has full knowledge about the behaviors and its teammate. Also, instead of having the two predators cooperate directly, we consider the case where they take turns trying to capture the prey, but both observe the results of the other's actions. Unfortunately, due to a defect, the teammate is not able to execute all of the behaviors that are available to the ad hoc agent, including the behavior with the best expected time to capture the prey. Therefore, there is a cost to the ad hoc team agent foregoing this best behavior in favor of another that will teach its teammate. The agents are still trying to maximize their shared rewards, but they take turns choosing behaviors to capture the prey. The ad hoc agent has complete information about the effectiveness of the behaviors, so it should help guide its naive teammate towards behaviors that are more effective. We consider the case where the teammate chooses behaviors greedily with respect to the sample means it has seen.

In this domain, the ad hoc team agent is the *teacher*, and it has perfect knowledge about its teammate, i.e. TeamK $= 1$. Also, its information about the environment is perfect in both the transition and reward distributions, i.e. EnvK $= (1, 1)$. Note that the other agent only has limited information about the environment. However, the reactivity of the domain depends on the number of episodes as well as the payoff distributions of the shared behaviors, but not the behavior that only the teacher can play. Similar to the scenario in Section 4.3, the challenge arises from the ad hoc agent needing to plan about the reactivity of its teammate. In addition, it must also consider how much information its teammate has, and how this affects the teammate's actions.

Therefore, the ad hoc team agent should reason about when it is useful to sacrifice choosing the behavior with the highest reward to teach its teammate about which behaviors it should be choosing. Close examination of this problem reveals that it can be modeled by a multi-armed bandit (MAB), such as that proposed by Stone and Kraus [17], where the different behaviors correspond to different arms of the bandit. In this setting, choosing a high level behavior to play for an episode corresponds to pulling the arm on the multi-armed bandit, and the length of the episode corresponds to the payoff of the arm. Stone and Kraus prove several interesting theoretical results about this formulation of ad hoc teamwork in a multi-arm bandit domain regardless of the payoff distributions of the arms, proving some theorems about when the ad hoc agent should and should not teach. Furthermore, they give efficient algorithms for handling cases where the payoff distributions are constrained.

From this research, we know that the ad hoc agent should consider playing behaviors other than the best one, which the teammate cannot play. In other words, it is advantageous for the ad hoc agent to teach its teammate despite the cost of teaching. Also, the ad hoc agent should never play the worst of the behaviors, even when the teammate's estimates of that behavior's quality is too high. Furthermore, with discrete distributions for the payoffs, Stone et al. give a polynomial time algorithm for calculating the optimal behavior for the ad hoc agent.

Consider the case in which the agents play on a 5x5 grid and must choose from three policies: the greedy, probabilistic destinations, and greedy probabilistic predators from Section 4.1. In this case, the teammate has non-deterministic actions and therefore cannot follow the purely greedy policy. These different policies give average capture times of 4.204, 4.272, and 4.911 respectively for a single predator capturing the prey, where a smaller time is better. The reactivity of this problem is 0.0342 if we consider histories of actions of length 100, and 0.128 for histories of length 10. Overall, a state is more reactive when the teacher can change the relative values of the arms' sample means, which happens when there are a small number of pulls that are far from the true arm means. When there are many pulls or the sample means closely match the true means, the reactivity is very low.

From Stone et al.'s work, we know that the ad hoc agent should consider sacrificing its reward from following the greedy policy to play other behaviors and teach its teammate. Also, we know that it should never play the greedy probabilistic behavior, even if its teammate thinks that this behavior is best.

5. CLASSIFYING EXISTING AD HOC RESEARCH DOMAINS

Section 4 introduces several ad hoc team problems and explains how they are described by the dimensions proposed in Section 3. We summarize those results here, and continue on to explore other domains used in previous ad hoc research. We then use these results to suggest new avenues for research into ad hoc teamwork.

5.1 Characteristics of the Pursuit Domain

A summary of the characteristics of variations of the pursuit domain is given in Table 2. Note that the reactivity of 2PS relies on the memory bound k of the agent (0.198 if $k = 1$), and in 2PT, it relies on the length of pull histories considered (0.0342 if the history size is 100 and 0.128 for length 10). For 4PKB and 4PUB, the values are affected by the size of the domain and the specific behaviors used by the teammates. Specifically, the reactivity is 0.00105 when for GP teammates on a 20x20 world, and 0.501 for TA teammates on a 5x5 world. The TeamK varies from 0.155 when cooperating with the teammates created by students on a 5x5 world to 0.807 for teammates drawn from {GR,GP,TA,PD} on a 20x20 world. From this table it becomes clear that research into ad hoc teamwork has focused mainly on the reactivity of the problems, with some approaches handling some uncertainty about the teammate behaviors. However, no ad hoc teamwork research thus far has handled any domains in which the environment is unknown. In addition, deviating slightly from the assumptions of either the 2PS or 2PT domains can render the theoretical results incorrect. Therefore, it is an important future direction to create general methods for handling problems with perfect knowledge about the teammates and environment, with varying reactivities. We hypothesize that the MCTS agent from Barrett et al. [1] should perform well in these domains, but this exploration remains open.

Domain	TeamK	EnvK	Reactivity
4PKB	1	(1,1)	0.00105–0.501
4PUB	0.155–0.807	(1,1)	0.00105–0.501
2PS	1	(1,1)	0.198
2PT	1	(1,1)	0.0342–0.118

Table 2: A summary of pursuit problems

5.2 Characteristics of Other Domains

Besides pursuit, there have been several other domains used in ad hoc teamwork research, though not always under the name of ad hoc teamwork. Although it is difficult to exactly calculate some of the dimensions without exact specifications of the domains, we estimate the values in Table 3.

Han et al. [9] explore using an agent to affect the collective behavior of a multi-agent system. Specifically, their work focused on adding a "shill" agent that was externally controlled, corresponding to the ad hoc team agent in our terminology. They then investigated using this agent to affect the behavior of groups of agents such as flocks of birds, directing the movement of the flock in a desired direction. They use a modified Boid model, in which each agent chooses its current heading by moving in the average direction of their neighbors located within a neighborhood of radius r. Table 3 summarizes how the domain is characterized along the dimensions. In this case, the shill agent knows its teammates' behavior and the environment, and the reactivity of its teammates depends on the number of agents and the size of the neighborhood. For these calculations, we discretize the actions into 10 degree bins. With 5 agents, the reactivity ranges from 0.880 when $r = 5$ to 0.0106 when $r = 1$, while with 100 agents, it ranges from 0.531 to 0.0732 for the same r values. This shows that while the reactivity can be large, in many settings the teammates are not strongly affected by the ad hoc agent's actions in the short term. However, Han et al.'s work shows that the long term effects of the ad hoc agent's actions are very influential, as the effects ripple out to the other agents.

Domain	TeamK	EnvK	Reactivity
Flocking control	1	(1,1)	0.0732–0.880
Cooperating with UTM-1 teammates	0	(1,1)	0
Cooperating with UTM-2 teammates	0	(1,1)	>0
Simulated pickup soccer	>0	(>0,1)	>0

Table 3: Estimates of other ad hoc team problems

More recently, Wu et al. [22] investigated ad hoc teamwork with few assumptions about the behaviors of the teammates. Their ad hoc agent plans using MCTS and uses biased adaptive play to predict the actions of teammates. Biased adaptive play can be used to estimate the policies of teammates from their previous actions. They test their agent on three domains: cooperative box pushing, meeting in a 3x3 grid, and multi-channel broadcast. They consider the case where the ad hoc agent knows the environment, but not its teammates. These teammates are referred to as unknown teammates (UTM), and two types of teammates are used in each domain: UTM-1 agents that follow a fixed set of actions and UTM-2 agents that try to play the optimal behavior but have partial observations. Along the dimensions, only the reactivity of the teammates vary between these three domains. However, the specifications of the UTM-2 agents is only that they act rationally with respect to partial observations of the system state, so it is not possible to calculate the exact values of the reactivity; it is only known that their reactivity is greater than 0. If the UTM-2 agents perform close to the rational behavior given full observations, it is expected that their reactivity is very high in these domains.

In the domain of simulated robot soccer, Bowling and McCracken [3] measure the performance of a few ad hoc agents, where each ad hoc agent is given a playbook that differs from that of its teammates. In this domain, the teammates implicitly assign the ad hoc agent a role, and then react to it as they would any teammate. This means that they react to the ad hoc agent's actions, i.e. the reactivity is greater than 0, but the extent of this reactivity is depends on their standard soccer behavior. In addition, the ad hoc agent knows a set of possible plays that may overlap with the plays that its teammates choose. It is expected that its knowledge of its teammates is fairly high, as effective soccer plays are similar, compared to random movement of the teammates, therefore TeamK is greater than 0. Although the ad hoc agent does not know ahead of time the noise caused by the simulation of the game or by the noise caused by the other team, it has reasonable expectations of the dynamics of the world, so TransK is greater than 0. In addition, it knows that scoring goals gives a positive reward, and giving up goals gives a negative reward, so RewardK is 1.

5.3 Characteristics of Future Research

Looking at how the existing research fits into the proposed dimensions gives us insight on directions for future investigation. Most research on ad hoc teams has focused on how teammates react to the ad hoc agent, as shown by the high levels of reactivity in existing domains. Little research has approached the problem of having low knowledge of the teammates, where the ad hoc agent must learn about its teammates to plan effectively. Wu et al.'s work [22] assumes that the ad hoc agent knows nothing about its teammates, but they focus on smaller domains. Barrett et al. [1] consider the idea that the ad hoc agent may start with some knowledge about its possible teammate, but must still learn about them by interacting with them. More research needs to be performed to investigate cases where the ad hoc agent knows little about its teammates. In many cases, agents can make some reasonable assumptions about the behavior of their teammates. Therefore, it is desirable to focus on ad hoc agents that cooperate with teammates starting with low, but nonzero information about their behaviors. In this case, the ad hoc agent must learn more about its teammates by interacting with them.

In addition most ad hoc teamwork research assumes that the ad hoc agent completely knows the transition dynamics of the environment as well as the short term rewards of actions. In other words, research into ad hoc teamwork has mainly focused on the difficulties of planning effectively with teammates, where the agent does not need to learn about the environment. On the other hand, many real world applications of ad hoc teamwork requires the agent to learn about its environment and adapt accordingly. In the case of search and rescue, robots must cooperate with previously unseen teammates, but they must also adjust to a noisy, changing environment. To perform effectively while exploring new environments, such as those encountered in exoplanet exploration, robots must learn about how their actions interact with the world and handle changes to their abilities caused by wear and tear. Therefore, future research in ad hoc teamwork should incorporate domains in which the dynamics begin unknown. This will create ad hoc agents that can trade off between exploring the environment, exploring interactions with their teammates, and exploiting their current knowledge. We believe that research in these areas is necessary to create robust, effective ad hoc agents.

6. RELATED WORK AND DISCUSSION

Aside from the ad hoc teamwork domains described in Section 5, some other research into ad hoc teams exists, such as Jones et al.'s [12] research into pickup teams working in the treasure hunt domain. This work assumes that the agents share a communication protocol that they use to bid on different roles. In addition, Knudson and Tumer [14] investigated ad hoc teams in a different framework. However, they assume that all agents in the domain adapt and that each agent is given a difference objective, which clearly specifies how an agent's actions affect its team's performance. Earlier research includes Brafman and Tennenholz's research [4] on agents performing a repeated joint task, where one agent attempts to teach a novice agent. A large body of research on coordinating multi-agent teams exists, specifying standardized protocols for communication and shared algorithms for coordination. These approaches include SharedPlans [8], STEAM [19], and GPGP [7].

Our work does not require any shared protocols, and does not assume that the teammates are adapting to the ad hoc team agent.

Modeling teammates is similar to the problem of opponent modeling, but it is generally safe to make stronger assumptions about teammates' behaviors. Therefore, the ad hoc agent does not need to consider the worst case scenario for every action; its teammates are trying to reach the same goal. The Workshop on Plan, Activity, and Intent Recognition (PAIR) and the Workshop on Applied Adversarial Reasoning and Risk Modeling (AARM) both have several papers relevant to applied opponent modeling, and there are also more theoretical approaches such as the AWESOME algorithm [6]. An interesting avenue for future work is to classify the much broader existing literature on opponent modeling along the same dimensions presented in this paper. These dimensions may aid in identifying approaches from opponent modeling literature that are likely to apply in corresponding ad hoc teamwork domains.

Isaacs performed seminal research on pursuit and evasion [10], and the problem was further explored by Benda et al. [2]. The pursuit domain has been well studied in multiagent research [18]. Most previous research focused on developing coordinating the predators before deploying them, rather than learning to adapt to unseen teammates. For example, MAPS [20] considers partially observable environments with the prey following sophisticated strategies, but requires a shared coordination algorithm. Other approaches focus on partial observability in continuous pursuit problems [11]. On the other hand, Chakraborty and Sen [5] investigate a pursuit scenario in which experienced agents attempt to teach novice predators, but require the agents to share a known training protocol. However, none of these methods are directly applicable to ad hoc teamwork.

7. CONCLUSION

This paper presents a set of dimensions for describing ad hoc team problems, and explains how these dimensions define the relationship among existing ad hoc team research studies. We show that reasoning about these dimensions aids in applying existing theoretical results to new problems.

The introduction of the dimensions describing the difficulties of ad hoc team problems raises several interesting avenues for future research. For example, by examining existing research in this light, it becomes clear that research thus far on ad hoc teams assumes that the adhoc agent knows the environment. Future work is needed on domains where the environment is unknown and the ad hoc agent must reason about the tradeoffs of exploration. Furthermore, domains where the ad hoc agent has less information about its teammates ought to be investigated. All of the results in this paper are in the context of the pursuit domain, broadly defined. Investigations of other domains, and whether existing algorithms apply in these domains is an important avenue for future research. From a high-level perspective, this research contributes towards understanding and solving the long-term challenge of creating robust, general ad hoc team agents.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at The University of Texas at Austin. LARG research is supported in part by grants from NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHWA (DTFH61-07-H-00030). Samuel Barrett is supported by an NDSEG fellowship.

8. REFERENCES

[1] S. Barrett, P. Stone, and S. Kraus. Empirical evaluation of ad hoc teamwork in the pursuit domain. In *AAMAS '11*, May 2011.

[2] M. Benda, V. Jagannathan, and R. Dodhiawala. On optimal cooperation of knowledge sources - An empirical investigation. Technical Report BCS–G2010–28, Boeing Advanced Technology Center, Boeing Computing Services, July 1986.

[3] M. Bowling and P. McCracken. Coordination and adaptation in impromptu teams. In *AAAI*, pages 53–58, 2005.

[4] R. I. Brafman and M. Tennenholtz. On partially controlled multi-agent systems. *JAIR*, 4:477–507, 1996.

[5] D. Chakraborty and S. Sen. Teaching new teammates. In *AAMAS '06*, pages 691–693, 2006.

[6] V. Conitzer and T. Sandholm. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Mach. Learn.*, 67, May 2007.

[7] K. S. Decker and V. R. Lesser. Designing a family of coordination algorithms. In *ICMAS '95*, pages 73–80, June 1995.

[8] B. Grosz and S. Kraus. Collaborative plans for complex group actions. *Artificial Intelligence*, 86:269–368, 1996.

[9] J. Han, M. Li, and L. Guo. Soft control on collective behavior of a group of autonomous agents by a shill agent. *Journal of Systems Science and Complexity*, 19:54–62, 2006.

[10] R. Isaacs. *Differential Games: A Mathematical Theory with Applications to Warfare and Pursuit, Control and Optimization*. Dover Publications, 1965.

[11] Y. Ishiwaka, T. Sato, and Y. Kakazu. An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning. *Robotics and Autonomous Systems*, 43(4):245 – 256, 2003.

[12] E. Jones, B. Browning, M. B. Dias, B. Argall, M. M. Veloso, and A. T. Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *ICRA*, pages 570 – 575, May 2006.

[13] T. Jung, D. Polani, and P. Stone. Empowerment for continuous agent-environment systems. Technical Report AI-10-03, The University of Texas at Austin Computer Science Department, 2010.

[14] M. Knudson and K. Tumer. Robot coordination with ad-hoc team formation. In *AAMAS '10*, pages 1441–1442, 2010.

[15] P. Stone, G. A. Kaminka, S. Kraus, and J. S. Rosenschein. Ad hoc autonomous agent teams: Collaboration without pre-coordination. In *AAAI '10*, July 2010.

[16] P. Stone, G. A. Kaminka, and J. S. Rosenschein. Leading a best-response teammate in an ad hoc team. In *AMEC*. November 2010.

[17] P. Stone and S. Kraus. To teach or not to teach? Decision making under uncertainty in ad hoc teams. In *AAMAS '10*, May 2010.

[18] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *Autonomous Robots*, 8(3):345–383, July 2000.

[19] M. Tambe. Towards flexible teamwork. *JAIR*, 7:81–124, 1997.

[20] C. Undeger and F. Polat. Multi-agent real-time pursuit. *AAMAS '10*, 21:69–107, July 2010.

[21] J. M. Vidal and E. H. Durfee. Recursive agent modeling using limited rationality. In *ICMAS*, 1995.

[22] F. Wu, S. Zilberstein, and X. Chen. Online planning for ad hoc autonomous agent teams. In *IJCAI*, 2011.

Modeling and Learning Synergy for Team Formation with Heterogeneous Agents

Somchaya Liemhetcharat
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
som@ri.cmu.edu

Manuela Veloso
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213, USA
veloso@cs.cmu.edu

ABSTRACT

The performance of a team at a task depends critically on the composition of its members. There is a notion of synergy in human teams that represents how well teams work together, and we are interested in modeling synergy in multi-agent teams. We focus on the problem of team formation, i.e., selecting a subset of a group of agents in order to perform a task, where each agent has its own capabilities, and the performance of a team of agents depends on the individual agent capabilities as well as the synergistic effects among the agents. We formally define synergy and how it can be computed using a synergy graph, where the distance between two agents in the graph correlates with how well they work together. We contribute a learning algorithm that learns a synergy graph from observations of the performance of subsets of the agents, and show that our learning algorithm is capable of learning good synergy graphs without prior knowledge of the interactions of the agents or their capabilities. We also contribute an algorithm to solve the team formation problem using the learned synergy graph, and experimentally show that the team formed by our algorithm outperforms a competing algorithm.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Capability, synergy, team formation, heterogeneous

1. INTRODUCTION

It is clear that the performance of a team, in terms of the outcome of the task, depends on the team composition. The term *synergy* is commonly used in human teams, and describes how well the team works together. We extend this notion of synergy from human teams to multi-agent teams, and seek to model and quantify it for effective team formation at a task.

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

Research in agent capabilities, which we describe in detail in the related work section, typically focus on single-agent capabilities, and not the interactions between multiple agents. Similarly, research in coalition formation typically does not seek to model how the value of a coalition is computed. In this paper, we introduce a model that captures synergy from the interactions among large groups of agents.

Concretely, we abstract the problem as finding the best subset of agents to complete a task, where each agent has its own set of capabilities. The performance of a team of agents at the task depends both on the capabilities of the agents, and the synergy among the members of the team. This model of performance among the agents is initially unknown, but observations of the performance of groups of agents can be made. From these observations, a model of synergy within the agents is learned, and the optimal subset of the agents is then selected for the task.

Formally, we define a synergy graph that models single-agent capabilities and the interactions among the agents. We then define pairwise synergy, i.e., how well a pair of agents will perform together, and define synergy in groups of agents. We then contribute an algorithm to find the optimal team to perform the task from a synergy graph, and an algorithm that learns a synergy graph from observations of interactions among small groups of agents. In our experiments, we show that the learned synergy graph matches closely to the hidden model that generated the observations, and the team formed by searching this learned synergy graph performs very well. In addition, we use a probabilistic robot capability model introduced in [8], and show that the synergy graph learned from observations of this probabilistic model leads to the formation of a team that outperforms the team selected by the ASyMTRe algorithm [8].

The format of our paper is as follows: in Sec. 2, we discuss related work and the differences with our work. In Sec. 3, we formally define the problem, and contribute our synergy graph model and the algorithm for team formation. In Sec. 4, we contribute an algorithm that learns a synergy graph based on observations of the performance of agents at the task. In Sec. 5, we describe our experiments and results, and we summarize our contributions in Sec. 6.

2. RELATED WORK

In heterogeneous teams, agents and robots have different capabilities. There has been a large amount of research in the task-allocation and role assignment domains [3], but the capabilities are typically binary, i.e., whether a robot is capable of performing a sub-task is usually due to its physical characteristics such as the presence of an arm. There has

also been some research in modeling capabilities as values, where higher values indicate better task performance [5], and with a Normal distribution to represent the uncertainty in the agents' performance [4]. However, while modeling capabilities of single agents and robots has been extensively studied, there has been limited work in modeling the capabilities of teams of agents, other than a sum of their individual capabilities, or a binary to represent whether the team can perform a sub-task. Pairwise capabilities between agents has been studied [7], and the coupling of robot capabilities to perform complex tasks using schemas with the ASyMTre algorithm [8], which we elaborate further below.

In the ASyMTRe algorithm, robot capabilities are modeled with schemas, that define inputs and outputs of information types (e.g., the global position of the robot) [8]. Each robot has a set of schemas with probabilities of success. The task is defined as a set of desired outputs, and a multi-robot team is formed to complete the task by creating a joint plan of the robots' schemas. Teams are ranked by a utility function that balances the probability of success and the cost of execution. ASyMTRe is an anytime algorithm that requires prior knowledge of the agent's capabilities and probabilities of success in order to rank potential teams, while our approach does not need such *a priori* information. Our approach models and learns the interactions between large groups of agents to form an effective team, and we compare the performance of our algorithm against ASyMTRe.

Coalition formation is a related field, where every possible subset of agents is given a value, and the goal is to partition the agents so as to maximize the sum of values. However, most of the research in the field has focused on how to achieve the best partitioning [9]. There has been some recent work in modeling how the value of a coalition is affected by externalities [1], but not how the value of a coalition is derived based on the composition of its members. Service and Adams recently applied coalition formation to solve task allocation, where the goal is to maximize the utility gained from completing tasks, and taking into account the resources/services that the agents can provide [10]. However, they use a market-based approach to solve the task allocation and do not model the synergistic effects across agents that have the same service. Our approach models varying quality in the capabilities of agents and how interactions in a team can amplify the results.

In the social network domain, there has been much research in selecting teams based on the interactions of agents in the social network graph. Lappas, Liu and Terzi studied how to find a team of experts that accomplish a goal while minimizing the communication cost in a social network graph [6]. Similarly, Dorn and Dustdar studied how to compose near-optimal expert teams by trading-off between coverage and communication cost [2]. We extend the use of a social network graph to model synergy between teams of agents, where the distance in the graph correlates with how well agents work together, and not the communication cost between the agents, and thus, the task performance is directly affected by the structure of the graph. Further, we contribute an algorithm to learn the structure of the graph from observations.

3. MODELING SYNERGY FOR TEAM FORMATION

Let $\mathcal{A}$ be a set of agents, and the task be T. Let the task T be divided into M independent sub-tasks, and let $F : 2^{\mathcal{A}} \rightarrow X$, where X is a M-dimensional random variable with unknown distribution. The function F represents the "world", so F is unknown but samples of $F(A)$ can be retrieved for agent teams $A \subseteq \mathcal{A}$, where the size of a team can vary from 1 to $|\mathcal{A}|$. A sample of $F(A)$ corresponds to the values attained by A at the M sub-tasks.

Let $V : \mathbb{R}^M \rightarrow \mathbb{R}$ be a value function that computes the overall value at the task T based on the values of the M sub-tasks. Examples of V are: $V_{sum}(X) = \sum_{m=1}^{M} X(m)$, $V_{max}(X) = \max_{m=1}^{M} X(m)$, and $V_{task}(X) = V_{sum}(X)$ iff $X(m) > \tau \, \forall \, m \in [1, M]$, and 0 otherwise, where $X(m)$ is the m^{th} component of X. V_{sum} and V_{max} are the summation and maximum functions, while V_{task} is a composite function that returns 0 if the performance of any sub-task is below a threshold τ, and is the summation otherwise.

The goal is to find a team of agents $A^* \subseteq \mathcal{A}$ such that $\forall \; A \subseteq \mathcal{A}, \; V(F(A^*)) \geq V(F(A))$, i.e., A^* receives the highest value at the task T.

3.1 Modeling Task-Based Relationships

In order to solve this problem of forming the optimal team to complete the task T, we create a model of F based on samples of $F(A)$. We assume that there is some task-based relationship between the agents that we can model. Research in the social network field use social graphs and communication costs to form effective teams [6, 2]; we model task-based relationships with a task-based network between the agents. As such, we form a task-based graph, where vertices are agents, and the edges represent the task-based relationship between the agents.

One method to model this relationship is with a fully-connected graph, where the weights of the edges represent how well agents work together (smaller numbers mean agents work better together, i.e., with lower cost). For example, Fig. 1 shows a 3-agent graph, where a_1 and a_2 work well together compared to other pairs.

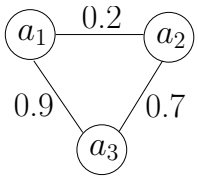


Figure 1: A fully-connected task-based graph where the task-based relationship between agents are represented by edge weights.

However, a fully-connected graph does not capture transitivity in the task-based relationship. For example, transitivity would mean that if a_1 works very well with a_2, and a_2 works very well with a_3, then a_1 should work well with a_3. To capture transitivity in the task-based relationship, we can use a connected graph (instead of a fully-connected one), where the minimum distance between agents in the graph is inversely correlated with their task-based relationship. Fig. 2 shows a modification from Fig. 1 where the edge $\{a_1, a_3\}$ has been removed, as an example that still preserves the distance between the agents.

In order to model the inverse correlation between the distance of agents and their task-based relationship, we introduce a weight function $w : \mathbb{R}^+ \rightarrow \mathbb{R}^+$, where $w(d(a_1, a_2))$ returns the task-based value of agents a_1 and a_2 based on

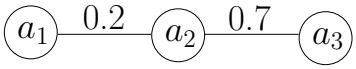


Figure 2: A connected task-based graph where the task-based relationship between agents is a function of the shortest distance between them.

the minimum distance between them in the graph (using the function d). Further, $w(d)$ is always positive and monotonically decreases as d increases. Intuitive examples of w are $w(d) = \frac{1}{d}$, and $w(d) = \exp(-\frac{d \ln 2}{h})$, which is an exponential decay function with half-life h.

As a simplifying assumption, we assume the edges in the task-based graph are unweighted (all edges have weight 1), and the weight function is still capable of fully capturing the task-based relationship.

3.2 Quantifying Performance at the Task

We want to model F based on samples of $F(A)$, and so far we have introduced a graphical model to capture the task-based relationships between agents. However, there is an innate capability of agents that is still unmodeled. For example, even if a_1 works equally well with a_2 and with a_3, the value $F(\{a_1, a_2\})$ may be consistently higher than $F(\{a_1, a_3\})$, if a_2 is "better" at the task than a_3.

As such, the graph structure alone is insufficient to completely model F. We thus add a value to each vertex. Although F returns an unknown distribution, we assume that it can be represented by M Normal distributions, where each Normal distribution models the agent's performance at a sub-task. Fig. 3 shows a 6-agent graph, where $M = 1$. We use Normal distributions because the single peak corresponds to the agent's average performance, and the symmetric spread corresponds to the agent's variability.

Now, we formally define a synergy graph:

Definition 1. A **synergy graph** S is a tuple $\{G_S, N_S\}$, such that $G_S = (V_S, E_S)$ is a connected graph, and N_S is set of Normal distributions, where:

- $v_a \in V_S$ is a vertex in G_S and represents an agent $a \in \mathcal{A}$,
- E_S are unweighted edges in G_S, and
- $N_a = (N_{a,1}, \ldots, N_{a,M}) \in N_S$ is a list of M Normal distributions, where $N_{a,m} \sim \mathcal{N}(\mu_{a,m}, \sigma^2_{a,m})$ is the capability of agent a at sub-task $m \in [1, M]$.

Using a synergy graph, we can compute the performance of a pair of agents:

Definition 2. The **pairwise synergy** $\mathbb{S}_2(a, a')$ between 2 agents $a, a' \in \mathcal{A}$ in a synergy graph S is a list of M Normal distributions, given by $w(d(v_a, v_{a'})) \cdot (N_a + N_{a'})$, where each component of N_a and $N_{a'}$ is summed independently, $d(v_a, v_{a'})$ is the shortest distance between the $v_a, v_{a'}$ in G_S, and $w(d)$ is a positive weight function that monotonically decreases as d increases.

Using this definition of synergy between a pair of agents, we define synergy within a group of agents, i.e., their task performance, below:

Definition 3. The **synergy** $\mathbb{S}(A)$ of a set of agents $A \subseteq \mathcal{A}$ in a synergy graph S is the average of the pairwise synergy of its components, i.e., $\frac{1}{\binom{|A|}{2}} \cdot \sum_{\{a,a'\} \in A} \mathbb{S}_2(a, a')$

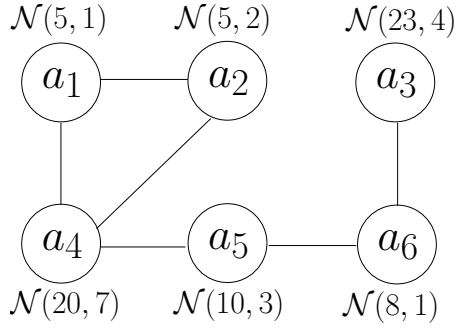


Figure 3: A synergy graph with 6 agents. Each vertex represents an agent, and the distance between vertices in the graph indicate how well agents work together. Agents have lists of Normal distributions that correspond to their capabilities at the M sub-tasks. In this example, $M = 1$.

Thus, $\mathbb{S}(A)$ returns $\{\mathcal{N}(\mu_{A,1}, \sigma^2_{A,1}), \ldots, \mathcal{N}(\mu_{A,M}, \sigma^2_{A,M})\}$, a list of M Normal distributions, indicating the task performance of the team $A \subseteq \mathcal{A}$. From Defs. 2 and 3:

$$\mu_{A,m} = \frac{1}{\binom{|A|}{2}} \sum_{\{a,a'\} \in A} w_{a,a'} \cdot (\mu_{a,m} + \mu_{a',m}) \quad (1)$$

$$\sigma^2_{A,m} = \frac{1}{\binom{|A|}{2}^2} \sum_{\{a,a'\} \in A} w^2_{a,a'} \cdot (\sigma^2_{a,m} + \sigma^2_{a',m}) \quad (2)$$

where $w_{a,a'} = w(d(v_a, v_{a'}))$ and $N_{a,m} \sim \mathcal{N}(\mu_{a,m}, \sigma^2_{a,m})$ are the agent capabilities at sub-task $m \in [1, M]$, assumed to be independent.

While the definition of synergy involves the summation of individual capabilities, it is weighted by the distance of agents in the synergy graph, and as such, the addition or removal of specific agents can have a large impact on the total score of a team. For example, from Fig. 3, suppose that $w(d) = \frac{1}{d}$. Then, the team $\{a_1, a_2\}$ has a mean score of 10, but the addition of a_3 lowers the mean to 8. Conversely, the team $\{a_1, a_2, a_4\}$ increases the mean to 20, even though the individual capability of a_4 is lower than a_3.

We defer the learning of synergy graphs from samples of F to a later section, and first describe how to find the best team $A^* \subseteq A$ given a synergy graph S.

3.3 Composing an Effective Team

In this section, we introduce an algorithm to approximate the optimal team composition for the task, in terms of the task performance, given a synergy graph. The goal is to find the optimal team $A^* \subseteq \mathcal{A}$ from a synergy graph S. We assume that the size of the optimal team (i.e., $n^* = |A^*|$) is known. This is a reasonable assumption, since the size of teams are typically limited by external factors, e.g., a cost budget, size restrictions. For example, the size of teams in sports is fixed, and also in tasks that require handing of a fixed number of devices, such as the operators of an ambulance. In addition, if n^* is unknown, then our approach can be run iteratively for increasing n, and then return the optimal team found across all n.

$\mathbb{S}$ calculates the list of M Normal distributions of the team's performance. In order to rank teams, each Normal distribution needs to be converted into a single number. To do so, we use the evaluation function introduced by us in [7], that balances the mean and variance of a Normal distribu-

Algorithm 1 Approximating the optimal team of size n

ApproxOptimalTeam(S, n, ρ)

1: $A \leftarrow$ `GenerateRandomTeam`$(\mathcal{A}, n)$
2: $\{N_{A,1}, \ldots, N_{A,M}\} \leftarrow \mathbb{S}(A)$
3: $v \leftarrow V($`Evaluate`$(N_{A,1}, \rho), \ldots,$ `Evaluate`$(N_{A,M}, \rho))$
4: **for** $k = 1$ to k_{max} **do**
5: $\quad A' \leftarrow$ `RandomTeamNeighbor`(A)
6: $\quad \{N_{A',1}, \ldots, N_{A',M}\} \leftarrow \mathbb{S}(A')$
7: $\quad v' \leftarrow V($`Evaluate`$(N_{A',1}, \rho), \ldots,$ `Evaluate`$(N_{A',M}, \rho))$
8: $\quad$ **if** `P`$(v, v',$ `Temp`$(k, k_{max})) >$ `random`$()$ **then**
9: $\quad\quad A \leftarrow A'$
10: $\quad\quad v \leftarrow v'$
11: **return** A

tion using a risk factor $\rho \in (0, 1)$, such that when $\rho > \frac{1}{2}$, distributions with higher variances attain higher values. As such, `Evaluate`$(N_{A,m}, \rho) = \mu_{A,m} + \sigma_{A,m} \cdot \Phi^{-1}(\rho)$, where Φ^{-1} is the inverse standard Normal cumulative distribution function. Thus, the M Normal distributions are converted into M real numbers, and the value function $V : \mathbb{R}^M \rightarrow \mathbb{R}$ is computes the final task value.

Algo. 1 finds an approximation of the optimal team of a given size n. A random team is first generated, where n agents are randomly chosen. Next, simulated annealing is performed to optimize the team configuration, where a neighbor of the current team is created by replacing one agent with another agent not currently in the team. Lines 4-10 of Algo. 1 implements simulated annealing, where `Temp` is a temperature schedule, `P`(v, v', t) returns a value between 0 and 1 given values v, v' and a temperature t.

Our team formation algorithm runs in $O(|A|)$ time if n^* is known. Otherwise, Algo. 1 is run for increasing n for a total runtime of $O(|A|^2)$. A brute-force algorithm would take $O(\binom{|A|}{n^*})$ if n^* is known, and $O(2^{|A|})$ otherwise.

4. LEARNING SYNERGY GRAPHS

The previous section formalizes synergy and defines a synergy graph, which captures how members of a heterogeneous team interact. In addition, an algorithm to find the optimal team for a task was presented. However, in order to quantify synergy within a team, we first need to learn a synergy graph. In this section, we contribute an algorithm that learns a synergy graph from observations of task performance of groups of agents.

The value function $F : 2^{\mathcal{A}} \rightarrow X$ is unknown, but samples of $F(A)$ can be obtained for $A \subseteq \mathcal{A}$. F can be likened to a black-box system, or the real world where the teams, i.e., $A \subseteq \mathcal{A}$, perform the task and the value of their performance at each of the M sub-tasks is observed.

Definition 4. An **observation** o_A is a list of M values corresponding to an observed overall performance of members $A \subseteq \mathcal{A}$ at the M sub-tasks, i.e., $o_A = F(A)$ for one sample of F. An **observation group** $O_A = \bigcup o_A$ is the set of all observations of A at the task.

Each observation group O_A contains all the observations of a unique group A. We assume that $\mathcal{A}$, the set of all agents, is known *a priori*. Otherwise, $\mathcal{A} = \bigcup_{O_A} A$. From these observation groups O_A, we define the observation set:

Definition 5. An **observation set** O is the set of all observation groups, i.e., $O = \bigcup \{O_A\}$.

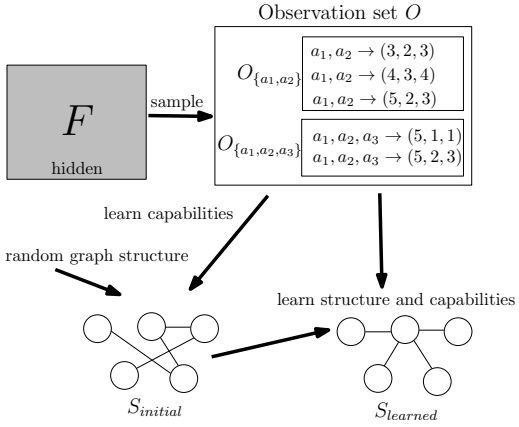


Figure 4: The process of learning from observations. Each observation is a list of M numbers, corresponding to each sub-task. The individual capabilities of agents in the synergy graphs are not shown.

Algorithm 2 Create a Synergy Graph from observations

CreateSynergyGraph$(O, \mathcal{A})$

1: $G \leftarrow$ `GenerateRandomGraph`$(\mathcal{A})$
2: $N \leftarrow$ `EstimateCapability`(O, G)
3: $S_{initial} \leftarrow \{G, N\}$
4: $S_{learned} \leftarrow S_{initial}$
5: $l \leftarrow$ `CalculateLogLikelihood`(O, S)
6: **for** $k = 1$ to k_{max} **do**
7: $\quad G' \leftarrow$ `RandomNeighbor`(G)
8: $\quad N' \leftarrow$ `EstimateCapability`(O, G')
9: $\quad S' \leftarrow \{G', N'\}$
10: $\quad l' \leftarrow$ `CalculateLogLikelihood`(O, S')
11: $\quad$ **if** `P`$(l, l',$ `Temp`$(k, k_{max})) >$ `random`$()$ **then**
12: $\quad\quad S_{learned} \leftarrow S'$
13: $\quad\quad l \leftarrow l'$
14: **return** $S_{learned}$

Fig. 4 illustrates the process of learning a synergy graph. The function F is sampled to obtain observations, which form the observation set O. An initial synergy graph $S_{initial}$ is created from a random graph structure and learned capabilities from the observation set. Subsequently an iterative algorithm is used to learn the synergy graph structure that best fits the observation set.

Algo. 2 explains this learning process in detail. First, `GenerateRandomGraph` creates a random graph G from agents $\mathcal{A}$, such that G is a connected graph, i.e., all vertices are connected through chains of edges. Next, `EstimateCapability` estimates the individual agent capabilities N, using the observation set O and graph G. The initial synergy graph S is then created from G and N, and the log-likelihood of the observations given S is calculated using `CalculateLogLikelihood`. Simulated annealing is then performed to converge on a synergy graph, where `P`(l, l', t) returns a value between 0 and 1 given values log-likelihoods l, l' and a temperature t. The log-likelihood of the observation set given a synergy graph is used as the maximizing criterion for simulated annealing, as the goal is to find a synergy graph that best matches the observations, i.e., is most likely to have produced the observations given its graph structure and individual capabilities. `RandomNeighbor`(G) is a function that takes an existing graph G, and either adds a new random

edge between two vertices, or removes an existing edge subject to the constraint that G remains a connected graph.

Algo. 3 estimates the individual agent capabilities, using the observation set O and graph G. Matrices $\mathcal{M}$ and b are created such that $\mathcal{M}x = b$, e.g., $\mathcal{M}_\mu\ x_\mu = b_\mu$, where $x_\mu = [\mu_{a_1}, \ldots, \mu_{a_{|\mathcal{A}|}}]^T$. Each row in $\mathcal{M}$ and b corresponds to an observation group O_A in O, using Eqns. 1 and 2. Each column in $\mathcal{M}$ corresponds to an agent in $\mathcal{A}$. A least squares solver is run to find x, which corresponds to the means and variances of the agent's capabilities at the m^{th} sub-task.

For example, suppose that $M = 1$ and $O_{\{a_1,a_2\}} = \{3, 4, 5\}$, i.e., the team $\{a_1, a_2\}$ was sampled 3 times using F and received value 3 for the first sample, 4 for the second, and 5 for the third. This observation group would form a row $[\alpha, \alpha, 0, \ldots]$ in $\mathcal{M}_\mu$, and a row $[\alpha^2, \alpha^2, 0, \ldots]$ in $\mathcal{M}_{\sigma^2}$, where $\alpha = w(d(v_{a_1}, v_{a_2}))$. b_μ and b_{σ^2} would then have a row with values 4 (the mean of the 3 observations) and 1 (the variance) respectively. Thus, each observation group creates a row in the $\mathcal{M}$ and b matrices, and a least-squares solver is used to solve for the means and variances of each agent.

`CalculateLogLikelihood` computed the log-likelihood of the observations, given a synergy graph S. In order to do so, for each observation group O_A in O, the synergy $\mathcal{N}(\mu_{A,m}, \sigma^2_{A,m})$ of the group $A \subseteq \mathcal{A}$ at the m^{th} sub-task is calculated using $\mathbb{S}$. The log-likelihood of each observed value in the observation O_A is then computed, and summed across all the observations and sub-tasks.

Overall, learning a synergy graph (Algo. 2) takes $O(r^3)$ time, where r is the number of observation groups, due to the least-squares solver in Algo. 3.

Algorithm 3 Estimate the individual agent capabilities

EstimateCapability(O, G)
Let $\mathcal{A} = \{a_1, \ldots, a_{|A|}\}$.
Let $G = (V_G, E_G)$, and $V_G = \{v_{a_j} : a_j \in \mathcal{A}\}$.
Let $O = \{O_{A_1}, \ldots, O_{A_r}\}$, where $A_i \subseteq \mathcal{A}$.
$\mathcal{M}_\mu \leftarrow \mathbf{0}_{r\times|\mathcal{A}|}$
$\mathcal{M}_{\sigma^2} \leftarrow \mathbf{0}_{r\times|\mathcal{A}|}$
$b_\mu \leftarrow \mathbf{0}_{r\times 1}$
$b_{\sigma^2} \leftarrow \mathbf{0}_{r\times 1}$
for $m = 1, \ldots, M$ **do**
  for all $O_{A_i} \in O$ **do**
    for all $a_j \in A_i$ **do**
      $\mathcal{M}_\mu(i,j) \leftarrow \frac{1}{\binom{|A_i|}{2}} \sum_{\{a_j,a\}\in A_i} w(d(v_{a_j}, v_a))$
      $\mathcal{M}_{\sigma^2}(i,j) \leftarrow \frac{1}{\binom{|A_i|}{2}^2} \sum_{\{a_j,a\}\in A_i} w(d(v_{a_j}, v_a))^2$
    // $o(m)$ is the m^{th} component of observation o
    $b_\mu(i) \leftarrow \frac{1}{|O_{A_i}|} \sum_{o\in O_{A_i}} o(m)$
    $b_{\sigma^2}(i) \leftarrow \frac{1}{|O_{A_i}|-1} \sum_{o\in O_{A_i}} (o(m) - b_\mu(i))^2$
  $means \leftarrow$ `LeastSquares`$(\mathcal{M}_\mu, b_\mu)$
  $variances \leftarrow$ `LeastSquares`$(\mathcal{M}_{\sigma^2}, b_{\sigma^2})$
  for all $a_j \in \mathcal{A}$ **do**
    $N_{a_j,m} \sim \mathcal{N}(means(j), variances(j))$
$N \leftarrow \{\}$
for all $a_j \in \mathcal{A}$ **do**
  $N \leftarrow N \cup \{(N_{a_j,1}, \ldots, N_{a_j,M})\}$
return N

5. EXPERIMENTS AND RESULTS

In order to test the efficacy of our synergy graph model, the learning algorithm to create synergy graphs from observations, and the performance of teams formed from the learned models, we split our experiments into two phases.

In the first phase, the hidden function F is set to be a synergy graph model, and our experiments are designed to show that the learned synergy graph matches the hidden synergy graph (in F) well. Further, we show that the teams formed from the learned synergy graph performs effectively compared to both the team formed from the hidden synergy graph, as well as the optimal team found by brute force.

In the second phase of our experiments, we compare our algorithms to the ASyMTRe algorithm [8]. The hidden function F follows the probabilistic model of robot capabilities in [8], i.e., F does not return a Normal distribution, and we learn a synergy graph model from observations of teams' performances. We then show that the team found from our learned synergy graph outperforms that of ASyMTRe.

5.1 F as a Synergy Graph

In order to create random synergy graphs for the function F, we first defined $|\mathcal{A}|$, the number of agents, and $p_{edge} \in (0, 1)$, the probability of an edge. We created a graph $G = (V_G, E_G)$ such that for each possible edge $e = \{v_1, v_2\}$, e was added into E_G if p_{edge} was greater than a random number uniformly generated in $[0, 1]$. Then, we checked that all agents in the graph were connected, otherwise the graph was discarded and re-generated. In our experiments below, we iterated between values of p_{edge} from $0.1, 0.2, \ldots, 0.9$ and collated the results across all p_{edge}.

Then, the simulator generated the agents' capabilities. In our first set of experiments, M, the number of sub-tasks, was set to 1. We generated $N_a \sim \mathcal{N}(\mu_a, \sigma_a^2) \in N_S$ such that $\mu_a \in [-\gamma, \gamma]$ and $\sigma_a^2 \in [0, \gamma]$, where γ is a multiplying factor, which we describe below. To generate the capabilities in the second set of experiments where $M > 1$, the M sub-tasks were first split among the agents, such that $\lceil \frac{|\mathcal{A}|}{M} \rceil$ agents were capable of performing each sub-task. Then, when generating the Normal distributions, if an agent a was capable of performing sub-task $m \in [1, M]$, then $N_{a,m} \sim \mathcal{N}(\mu_{a,m}, \sigma^2_{a,m})$, and $N_{a,m} \in N_a \in N_S$ such that $\mu_{a,m} \in (\frac{\gamma}{2}, \frac{3\gamma}{2})$, and $\sigma^2_{a,m} \in (0, \gamma)$. Otherwise, the distribution $N_{a,m} \sim \mathcal{N}(0, \epsilon)$ for some small ϵ.

We created 2 weight functions w, $w_{frac}(d) = \frac{1}{d}$ and $w_{decay} = \exp(-\frac{d \ln 2}{h})$, where d is the distance between 2 agents in the graph, and h is the half-life of the exponential decay function. The two weight functions were selected to demonstrate that the algorithms' performance is similar regardless of the weight function, and because both w_{frac} and w_{decay} were intuitive and easy to understand. For the experiments in this paper, we set $h = 3$, since $|\mathcal{A}|$ was set to be at most 10, so the weight between agents would have a similar range for both functions.

γ, the multiplying factor, affects how the performance of the agents are affected by the weight functions. For example, a weight of $\frac{1}{2}$ reduces a capability of 4 to 2 (a difference of 2 units), but reduces 40 to 20 (a much larger difference of 20), which could have effects on the learning algorithm. Thus, we varied γ in our learning experiments to study the effect of the range of utilities on the performance of our algorithm.

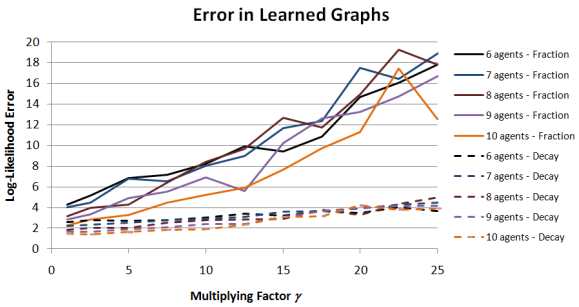


Figure 5: The error in the learned graph with varying number of agents and both weight functions.

5.1.1 Learning Synergy Graphs from Observations

In Sec. 4, we described the algorithm used to learn a synergy graph from observations. We first generated a synergy graph S_{true} using the method described above, and then generated a training observation set O_{train}, with sets of 2 and 3 agents, i.e., $\forall\, O_A \in O_{train}$, $2 \leq |A| \leq 3$. We generated 100 data points from each pair/triple, and modeled a synergy graph $S_{learned}$ from the data.

Then, to test how well our algorithm learns the synergy graph, we generated a test observation set O_{test} using combinations of 4 or greater agents, i.e., $\forall\, O_A \in O_{test}$, $|A| \geq 4$, that had 1000 observations in total. We then measured the difference in log-likelihoods between the hidden and learned synergy graphs, i.e., $LL(O_{test}|S_{true}) - LL(O_{test}|S_{learned})$, where $LL(O|S) =$ `CalculateLogLikelihood`(O, S). A low log-likelihood difference indicates that the synergy graph is as likely as the true graph to have produced the observations. We compared this difference in log-likelihood versus the initial graph used in the learning algorithm, $S_{initial}$, that had random edges but learned agent capabilities, to observe if the graph structure has an effect on the log-likelihoods.

We first ran experiments where there was a single task ($M = 1$) and agents had heterogeneous levels of performance with regards to the task. Fig. 5 shows the log-likelihood differences of the learned synergy graphs with the 2 weight functions compared to the hidden synergy graph, and varying the number of agents from 6 to 10. Varying the number of agents does not affect the log-likelihood error much — the weight function and the multiplier γ have greater effects.

Figs. 6 and 7 shows the log-likelihood difference of the learned synergy graphs and the initial synergy graphs when $|\mathcal{A}| = 10$. The learned synergy graphs are much closer to the true synergy graphs, i.e., the difference in log-likelihood is close to 0 and orders of magnitude lower than the initial synergy graphs. The error in log-likelihood increases as γ, the multiplying factor, increases, especially for $S_{initial}$, and shows that γ affects the difficulty of the learning problem (seen from the errors of $S_{initial}$), but our learning algorithm is capable of significantly reducing this error in $S_{learned}$. Furthermore, the observation set used for learning only included pairs and triples of agents, but the learned graph had a low log-likelihood difference when testing against data of teams comprising 4 or more agents, which shows that the structure of the learned graph and the individual agent capabilities match the hidden synergy graph well.

The next set of experiments were run where the task was composed of $M > 1$ sub-tasks, and each sub-task had a number of agents that were capable of performing it. We

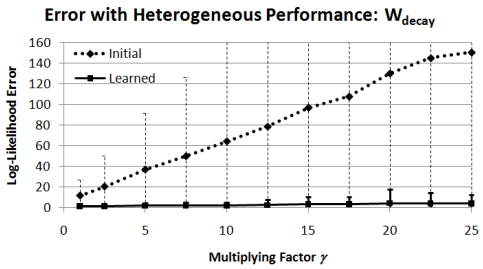


Figure 6: The error in the learned synergy graph of 10 agents with heterogeneous task performance, using the weight function $w_{decay}(d) = \exp(-\frac{d \ln 2}{3})$, compared with the initial graph used by the learning algorithm, with random structure but learned agent capabilities.

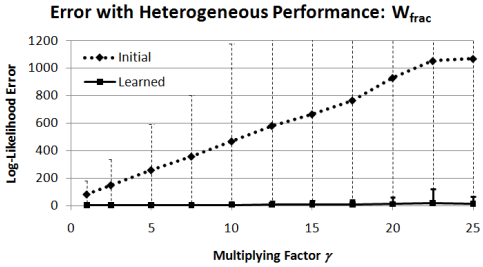


Figure 7: The errors in synergy graphs of 10 agents with heterogeneous task performance, using the weight function $w_{frac}(d) = \frac{1}{d}$.

used w_{decay} as the weight function, set $|\mathcal{A}| = 10$, and varied M from 2 to 5. Fig. 8 shows the log-likelihood differences between the $S_{learned}$ and the S_{true}, as compared to $S_{initial}$ and S_{true}. The error in the learned graphs are half or less than that of the initial graphs, which demonstrates the efficacy of our learning algorithm, using only data of 2 and 3 agents and being tested on larger groups of agents. However, while the error in log-likelihood of $S_{learned}$ remains mostly flat compared to γ, the error increases as M increases, which shows that an increase in the number of sub-tasks has a large impact in the quality of the learned synergy graph. The advantage of w_{decay} over w_{frac} should be general, because the decay function decreases less abruptly as distance increases.

5.1.2 Measuring Team Performance

The goal is to find the optimal team to perform the task, and we use `ApproxOptimalTeam` (Algo. 1) on the learned synergy graphs $S_{learned}$. The performance of this set of agents is then computed with F in the hidden synergy graph S_{true}, and compared against the best and worst possible combinations of agents. For example, if the set of agents A' is selected from $S_{learned}$, then the value of A' is computed on S_{true}.

We did two sets of experiments: when $M = 1$ and when $M > 1$. In the first set, we varied $|\mathcal{A}|$, the number of agents in the synergy graph, from 6 to 10, and the algorithm picked the best 5 agents. In the second set, we fixed $|\mathcal{A}|$ to 10 and varied M, the number of sub-tasks, from 2 to 5. In both cases, $\gamma = 1$, and $\rho = 0.75$. Tables. 1 and 2 show the score of the picked agents on a scale of 0 to 100 (where 0 denotes the worst possible combination, and 100 is the optimal team), and the performance of the selected team on the hidden

Figure 8: The error in the initial and learned synergy graphs of 6 agents where some sub-tasks could only be completed by some agents, using the weight function w_{decay}.

	Learned Graph	Hidden Graph
6 agents	99.90 ± 0.73	100.00 ± 0
7 agents	99.89 ± 0.45	100.00 ± 0.05
8 agents	99.94 ± 0.32	100.00 ± 0
9 agents	99.96 ± 0.28	100.00 ± 0
10 agents	99.95 ± 0.36	99.99 ± 0.12

Table 1: Score (%) of agents with 1 sub-task.

	Learned Graph	Hidden Graph
2 sub-tasks	99.64 ± 0.77	99.96 ± 0.40
3 sub-tasks	99.57 ± 3.43	99.97 ± 0.47
4 sub-tasks	98.79 ± 9.95	99.97 ± 0.38
5 sub-tasks	89.65 ± 30.26	99.97 ± 0.25

Table 2: Score (%) of agents: 10 agents with varying number of sub-tasks.

graph S_{true}. The worst and optimal teams were discovered by iterating through all possible combinations of agents and computing their value. It is remarkable that our algorithm finds a team that obtains a score of at least 89.65%, and has a similar score when the algorithm is run on the hidden graph, and thus shows that the learned synergy graphs are in fact very close to hidden synergy graphs that were used to generate the observation sets, and that `ApproxOptimalTeam` can be used to find effective teams.

5.2 F as a Probabilistic Function

In this section, F was no longer a hidden synergy graph model. Instead, we used the robot capability model of Parker and Tang [8], where every robot has a subset of actions that it can perform, each with a probability of success. These actions are then chained across robots to produce the desired output, again with some probability of success. Fig. 9 shows the capabilities of 3 agents and how the actions are chained together to produce the desired outcome. Since each agent has a subset of the actions, different subsets of agents will have different results. In our experiments, we varied the number of agents from 4 to 10 and randomly picked their capabilities in each trial — each agent had a 0.7 chance of being able to perform each action, and the probability of success of the action was uniformly sampled from $[0.1, 0.9]$.

F, the function of the performance of a team of agents, was calculated based on the cost of executing the actions and the reward achieved by generating the output. The cost of attempting actions 1, 2 and 3 were 30, 10 and 15 respectively, and the cost of attempting to generate the output from action 1 was 10 and 15 from the combination of action 2 and 3. When the output was achieved successfully, a reward of 100 was given. The values of costs and reward were arbitrarily chosen, but further experiments with different values yielded similar results, so we present these results in this paper. In each trial, every agent would attempt to execute its actions, and if they were successful, the output was also attempted to be generated. Thus, F had a probability density function (pdf) that depended on the agent capabilities — this pdf was not Normally distributed in general. We were interested to find out how accurately we could learn a synergy graph to model F even in such a situation.

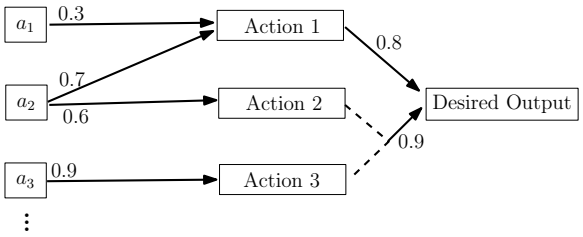


Figure 9: The model of robot capabilities introduced by Parker and Tang [8]. The numbers indicate probabilities of success, and the dashed lines out of actions 2 and 3 indicate that both are required to trigger the desired output.

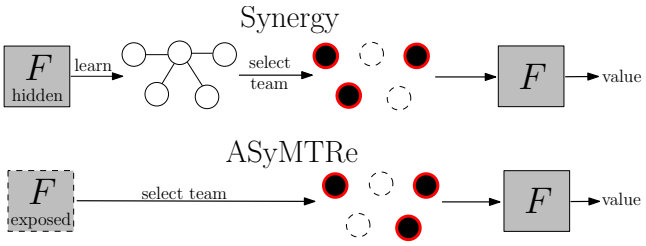


Figure 10: The experimental process to compare our synergy algorithm against the ASyMTRe algorithm.

Fig. 10 shows the experimental process. The hidden function F was used to generate observations of subsets of 2 and 3 agents, and then a synergy graph model is learned (Algo. 2). A team is then selected using the learned synergy graph (Algo. 1), and the value of the team is computed using F. To attain results for ASyMTRe [8], the probabilities of success and costs of actions in F were exposed, and the heuristic to rank teams in [8] was used. The ASyMTRe algorithm is an anytime algorithm, but for our experiments, we ran it to completion so that the optimal team with respect to the authors' heuristic was chosen. The values of the selected teams were then compared to the maximum and minimum team values, which were attained by performing a brute-force search of all possible combinations of agents in F, and thus scaling the results of the synergy and ASyMTRe algorithms to be between 0 and 1.

The ranking heuristic in the ASyMTRe algorithm has a factor $p \in [0, 1]$ that balances between the probability of success of performing an action versus the cost of the action. For our experiments, we varied p from 0 to 1 at 0.1 intervals, and collated the results. Similarly, the synergy algorithm uses the risk factor $\rho \in (0, 1)$; we varied ρ from 0.1

# of agents	Synergy	ASyMTRe
4	95 ± 17	64 ± 34
5	95 ± 14	64 ± 33
6	96 ± 10	63 ± 31
7	97 ± 8	60 ± 29
8	93 ± 7	59 ± 27
9	97 ± 7	59 ± 25
10	96 ± 8	63 ± 27

Table 3: Score (%) of teams composed by our synergy algorithm versus the ASyMTRe algorithm.

to 0.9 at 0.1 intervals and collated the results. The results were collated across p and ρ since the values were consistent and had little effect on the performance of the algorithms in general. We varied $|\mathcal{A}|$, the number of agents, from 4 to 10, and picked teams of sizes 2 to $|\mathcal{A}| - 1$. For a given size of $|\mathcal{A}|$, we performed 30 trials for each team size.

Table 3 shows the scores of the two algorithms. Across all number of agents, our synergy algorithm outperforms the ASyMTRe algorithm in terms of the performance of the team selected, even though the function F is hidden to the synergy algorithm but exposed for the ASyMTRe algorithm. The ASyMTRe algorithm finds teams that score around 60% of the optimal while the synergy algorithm forms teams that score above 90%. This significant difference is due to a number of reasons: firstly, the ASyMTRe algorithm was designed to also plan the agents' actions, i.e., which actions each agent should perform in order to complete the task. Secondly, the ASyMTRe typically plans for a set number of outputs (e.g., find a team to produce 2 outputs), but in our experiments the heuristic was used to find a team that produces as much output as possible. We compared our synergy algorithm to ASyMTRe as it is a well-known algorithm for multi-robot team formation and coordination that exploits heterogeneity in the agents to maximize task performance.

6. CONCLUSIONS

We are interested in team formation, where heterogeneous agents of varying capabilities are put together to complete a task. The interactions between these agents are initially unknown, and the goal is to select a subset of these agents such that the task performance is maximized.

We formally defined a synergy graph, where an agent's capability is represented by a list of Normal distributions, and the task-based relationship between agents are modeled by the distance between them in a graph. We then formally defined how pairwise synergy can be computed using a synergy graph, and extended the definition of synergy to include groups of any number of agents. We then presented an algorithm to approximate the optimal team given a synergy graph. Next, we contributed an algorithm that learns a synergy graph from observations of the performance of groups of agents, without any prior information about the agents' capabilities or the interactions among them. While we used simulated annealing in our team formation and learning algorithms, we believe that other approximation techniques would have similar performance.

In our experiments, we used 2 weight functions to weight the synergy of agents based on their distance in the graph. Using only observations of pairs and triples of agents, we showed that our learning algorithm is capable of learning the structure of task-based interactions and the capabilities of the agents as compared to the initial synergy graph where the observations were generated from, using both weight functions. This is a significant contribution as it shows that our learning algorithm does not need to observe all combinations of agent interactions in order to learn the synergy model, and is robust to different weight functions. Furthermore, we used a probabilistic model of agent capabilities to determine task performance, and compared our synergy algorithm with the ASyMTRe algorithm, and showed that even though the hidden model was not Normally distributed, and our algorithm does not have *a priori* knowledge of the agents' capabilities and probabilities of success of their actions while ASyMTRe has full knowledge, we are able to form teams that perform much better.

Acknowledgments

This work was partially supported by the Air Force Research Laboratory under grant no. FA87501020165, and the Agency for Science, Technology, and Research (A*STAR), Singapore. 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 any sponsoring institution, the U.S. government or any other entity.

7. REFERENCES

[1] B. Banerjee and L. Kraemer. Coalition structure generation in multi-agent systems with mixed externalities. In *Proc. 10th Int. Conf. Autonomous Agents and Multiagent Systems*, pages 175–182, 2010.

[2] C. Dorn and S. Dustdar. Composing near-optimal expert teams: A trade-off between skills and connectivity. In *Proc. Int. Conf. Cooperative Information Systems*, pages 472–489, 2010.

[3] B. P. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *Int. J. Robotics Research*, 23(9):939–954, 2004.

[4] C. Guttmann. Making allocations collectively: Iterative group decision making under uncertainty. In *Proc. 6th German Conf. Multiagent System Technologies*, pages 73–85, 2008.

[5] L. He and T. R. Ioerger. A quantitative model of capabilities in multi-agent systems. In *Proc. Int. Conf. Artificial Intelligence*, pages 730–736, 2003.

[6] T. Lappas, K. Liu, and E. Terzi. Finding a Team of Experts in Social Networks. In *Proc. Int. Conf. Knowledge Discovery and Data Mining*, pages 467–476, 2009.

[7] S. Liemhetcharat and M. Veloso. Mutual state capability-based role assignment model (extended abstract). In *Proc. 9th Int. Conf. Autonomous Agents and Multiagent Systems*, pages 1509–1510, 2010.

[8] L. Parker and F. Tang. Building multirobot coalitions through automated task solution synthesis. *Proc. IEEE*, 94(7):1289–1305, 2006.

[9] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohme. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111:209–238, 1999.

[10] T. Service and J. Adams. Coalition formation for task allocation: theory and algorithms. *Autonomous Agents and Multi-Agent Systems*, 22:225–248, 2011.

Session 1C
Learning I

V-MAX: Tempered Optimism for Better PAC Reinforcement Learning

Karun Rao
Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands
karunrao97@gmail.com

Shimon Whiteson
Informatics Institute
University of Amsterdam
Amsterdam, The Netherlands
s.a.whiteson@uva.nl

ABSTRACT

Recent advances in reinforcement learning have yielded several *PAC-MDP* algorithms that, using the principle of optimism in the face of uncertainty, are guaranteed to act near-optimally with high probability on all but a polynomial number of samples. Unfortunately, many of these algorithms, such as *R-MAX*, perform poorly in practice because their initial exploration in each state, before the associated model parameters have been learned with confidence, is random. Others, such as *Model-Based Interval Estimation* (MBIE) have weaker sample complexity bounds and require careful parameter tuning. This paper proposes a new PAC-MDP algorithm called *V-MAX* designed to address these problems. By restricting its optimism to future visits, V-MAX can exploit its experience early in learning and thus obtain more cumulative reward than R-MAX. Furthermore, doing so does not compromise the quality of exploration, as we prove bounds on the sample complexity of V-MAX that are identical to those of R-MAX. Finally, we present empirical results in two domains demonstrating that V-MAX can substantially outperform R-MAX and match or outperform MBIE while being easier to tune, as its performance is invariant to conservative choices of its primary parameter.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence—Learning

General Terms

Algorithms

Keywords

Reinforcement learning, Sample complexity

1. INTRODUCTION

In *reinforcement learning* (RL) [17], an agent must learn an optimal *policy* for maximising its expected long-term reward in an initially unknown *Markov decision process* (MDP) [1]. Since a wide range of realistic problems, from game playing to robot control, can be naturally formulated as MDPs, effective reinforcement-learning algorithms are critical to the development of intelligent agents.

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

A central challenge in RL is how best to balance *exploration*, in which the agent tries various actions to learn about their effects, and *exploitation*, in which it uses what it has already learned to select actions that maximise expected return. Traditional RL algorithms such as *Q-Learning* [21] rely on ad-hoc exploration mechanisms that ensure each state-action pair is experienced infinitely often. As a result, the convergence of such algorithms to the optimal policy is guaranteed only in the limit.

Fortunately, methods have recently been developed that explore more efficiently and thus obtain guarantees based on the *probably approximately correct* (PAC) [20] framework. Instead of converging to the optimal policy, PAC-MDP algorithms are guaranteed to act near-optimally with high probability on all but a polynomial number of samples.

R-MAX [2], the most well known PAC-MDP method, formalises the principle of *optimism in the face of uncertainty*. If n, the number of times a state-action pair has been visited, is less than a threshold m, it is assumed to have maximal value. Planning on the resulting model yields a policy that either leads the agent to unfamiliar state-action pairs or exploits familiar ones of high value.

Despite its PAC guarantees, R-MAX often performs poorly in practice because its initial exploration is random: until a state-action pair has been visited m times, the agent's experience with it is ignored. *Model-Based Interval Estimation* (MBIE) [16] avoids this problem by computing confidence intervals that quantify the agent's optimism. While MBIE has been shown to outperform R-MAX empirically, the bounds on its sample complexity are not as strong [16].

This paper describes *V-MAX*, a novel PAC-MDP method designed to overcome the weaknesses of both R-MAX and MBIE. Like R-MAX, V-MAX is optimistic about state-action pairs experienced fewer than m times. However, like MBIE, its optimism is tempered by experience. Rather than assuming such state-action pairs have maximal value, it assumes only that the remaining $m - n$ visits will yield maximal return. Thus, its estimate of the state-action pair's value is a weighted average of the expected return based on the n visits and that of the $m - n$ optimistic future visits.

In this way, V-MAX can exploit its experience early in learning and thus obtain more cumulative reward than R-MAX. Furthermore, this additional exploitation does not make exploration less efficient. On the contrary, we prove bounds on the sample complexity of V-MAX that are identical to those of R-MAX. This result shows for the first time that it is possible to employ tempered optimism like that of MBIE but without compromising the resulting PAC bounds.

This paper also presents empirical results comparing the performance of V-MAX to MBIE, R-MAX, and MoR-MAX, another PAC-MDP method, on two tasks: a standard benchmark task from the PAC-MDP literature and a new task containing multiple local optima that is designed to be challenging for PAC-MDP methods. The results on both tasks demonstrate that V-MAX can substantially outperform both R-MAX and MoR-MAX. They also demonstrate that V-MAX, unlike MBIE, performs well in the presence of multiple local optima. Even when such local optima are absent, V-MAX matches the performance of MBIE and in both cases proves substantially easier to tune, as its performance is invariant for conservative choices of m.

The rest of this paper is organized as follows. Section 2 provides background on RL, PAC, and PAC-MDP methods. Section 3 presents V-MAX and a proof of its sample complexity. Section 4 describes our experimental results while Section 5 concludes and suggests directions for future work.

2. BACKGROUND

A Markov decision process can be described as a five-tuple (S, A, R, T, γ), where S is the state space, A is the action space, $R(s,a)$ is the reward function describing the expected reward given state s and action a, $T(s,a,s')$ is the transition function describing the probability of arriving in state s' given s and a, and $0 \leq \gamma < 1$ is the discount factor used when summing an infinite sequence of rewards. An agent's policy $\pi : S \to A$ specifies what action to take in each state. An optimal policy π^* maximises the expected γ-discounted long-term cumulative reward, also known as the *expected discounted return* [17].

If both the reward and transition functions are known, then an agent can compute π^* using a planning method such as *value iteration* (VI) [1]. VI computes the *optimal action-value function* Q^* by iteratively solving the *Bellman optimality equation* (1), until for each state-action pair (s,a), subsequent computations of $Q^*(s,a)$ yield a difference less than some tolerance ϵ. Given Q^*, an optimal policy can be easily derived: $\pi^*(s) = \text{argmax}_{a \in A} Q^*(s,a)$.

$$Q^*(s,a) \leftarrow R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') \max_{a' \in A} Q^*(s',a') \quad (1)$$

If the reward and/or transition functions are unknown, then the agent can learn Q^* from experience. *Model-based* RL methods do so indirectly by learning $\hat{R}$ and $\hat{T}$, maximum likelihood estimates of the reward and transition functions, and planning on the resulting MDP: $(S, A, \hat{R}, \hat{T}, \gamma)$.

Learning $\hat{R}$ and $\hat{T}$ is complicated by the need to explore the environment to collect samples. If the agent explores too little, it may not learn the dynamics of the MDP accurately. If it explores too much, it may not accumulate enough reward. While it is possible in principle to compute a Bayes-optimal exploration strategy, doing so is typically intractable [3]. As a result, ad-hoc strategies such as ϵ-*greedy* exploration are often used in practice. The probably approximately correct (PAC) learning framework offers a middle ground: tractable algorithms that, while not Bayes-optimal, have upper bounds on their *sample complexity*:

Definition 1. For any fixed $\epsilon > 0$, the *sample complexity of exploration* of an algorithm $\mathcal{A}$ is the number of timesteps t such that the policy at time t, $\mathcal{A}_t$, satisfies $V^{\mathcal{A}_t}(s_t) < V^*(s_t) - \epsilon$ [8].

A PAC-MDP method is one that is guaranteed to have, with high probability, a polynomial sample complexity:

Definition 2. An algorithm $\mathcal{A}$ is *PAC-MDP* (Probably Approximately Correct in Markov Decision Processes) if, for any $\epsilon > 0$ and $0 < \delta < 1$, the sample complexity of $\mathcal{A}$ is less than some polynomial in the relevant quantities $(|S|, |A|, R_{\max}, 1/\epsilon, 1/\delta, 1/(1-\gamma))$ for any MDP M, with probability at least $1 - \delta$, where $R_{\max}$ is an upper bound on the reward function of M [13].[1]

R-MAX [2], described in Algorithm 1, is the simplest and most well known PAC-MDP algorithm. In addition to S, A, γ, and $R_{\max}$, it takes as input two parameters: m, the number of times each state-action pair must be experienced before $\hat{R}$ and $\hat{T}$ are considered near-accurate, and ϵ_1, the accuracy required from VI during the planning step.

Algorithm 1: R-MAX

Input: S, A, γ, m, ϵ_1, $R_{\max}$

$\bar{S} \leftarrow S \cup \{z\}$, where z is an arbitrary fictitious state
foreach $(s,a) \in \bar{S} \times A$ **do**
  $n(s,a) \leftarrow 0$
  $r(s,a) \leftarrow 0$
  $\tilde{Q}(s,a) \leftarrow R_{\max}/(1-\gamma)$
  $\tilde{R}(s,a) \leftarrow R_{\max}$
  foreach $s' \in S$ **do**
    $n(s,a,s') \leftarrow 0$
    $\tilde{T}(s,a,s') \leftarrow 0$
  end
  $n(s,a,z) \leftarrow 0$
  $\tilde{T}(s,a,z) \leftarrow 1$
end
for $t = 1, 2, 3, \ldots$ **do**
  Observe current state s
  Execute action $a := \text{argmax}_{a' \in A} \tilde{Q}(s,a')$
  Observe immediate reward r and next state s'
  if $n(s,a) < m$ **then**
    $n(s,a) \leftarrow n(s,a) + 1$
    $r(s,a) \leftarrow r(s,a) + r$
    $n(s,a,s') \leftarrow n(s,a,s') + 1$
    if $n(s,a) = m$ **then**
      $\tilde{R}(s,a) \leftarrow r(s,a)/m$
      foreach $s'' \in \bar{S}$ **do** $\tilde{T}(s,a,s'') \leftarrow n(s,a,s'')/m$
      $\tilde{Q} \leftarrow$ Solve $(\bar{S}, A, \tilde{R}, \tilde{T}, \gamma, \epsilon_1)$ using VI
    end
  end
end

R-MAX adds a fictitious maximally rewarding state z to the MDP and initially assumes that all state-action pairs (s,a) (including all (z,a)) yield the maximum reward $R_{\max}$ and transition with probability 1 to z.[2] We use $\tilde{R}$ and $\tilde{T}$ to denote these initially optimistic reward and transition functions. In addition, $\tilde{Q}$, the optimistic value function, is initialized to the maximum possible value, i.e., $\tilde{Q}(s,a) = V_{\max} = R_{\max}/(1-\gamma)$ for all s and a. Once some (s,a) has been experienced m times, $\tilde{R}(s,a)$ and $\tilde{T}(s,a)$ are set to $\hat{R}(s,a)$ and $\hat{T}(s,a)$, respectively, and the $\tilde{Q}$-values are updated with VI. Thus, R-MAX automatically explores state-action pairs that it is uncertain about and exploits otherwise. Once some (s,a) has been experienced m times, R-MAX assumes the

[1]This definition is slightly modified in that it allows the bounds to also depend on $R_{\max}$ since, unlike [13], we do not assume that $R_{\max} = 1$. In addition, we do not consider the space and computational complexity.

[2]We use z for ease of comparison to V-MAX, but R-MAX can also be implemented by initialising all state-action pairs to have self-transitions with probability 1.

reward and transition functions for (s,a) are near-accurate and stops learning them. The tightest known upper bounds on R-MAX's sample complexity, due to [13], are:[3]

$$O\left(\frac{|S||A|R_{\max}^3}{\epsilon^3(1-\gamma)^6}\Big(|S|+\ln\frac{|S||A|}{\delta}\Big)\ln\frac{1}{\delta}\ln\frac{R_{\max}}{\epsilon(1-\gamma)}\right).$$

Modified R-MAX (MoR-MAX) [19] is similar to R-MAX except that, once a state-action pair (s,a) has been experienced m times, it restarts the sample collection for (s,a). For every m such samples that MoR-MAX collects, it creates a trial model consisting of the reward and transition functions based on the m most recent samples of (s,a). It then uses VI to compute $\tilde{Q}'$ based on the trial model and, if $\tilde{Q}'(s,a) \leq \tilde{Q}(s,a)$, then it replaces the reward and transition functions for (s,a) with those of the trial model. In this way, MoR-MAX continues learning throughout the agent's lifetime, unlike R-MAX. Note that each time MoR-MAX replaces the reward and transition functions using m new samples, the previous m samples are discarded. With probability $1-\delta$, the sample complexity of MoR-MAX is:

$$O\left(\frac{|S||A|R_{\max}^2}{\epsilon^2(1-\gamma)^6}\ln\frac{|S||A|R_{\max}}{\delta\epsilon(1-\gamma)}\ln^2\frac{R_{\max}}{\epsilon(1-\gamma)}\right).$$

Ignoring log factors, the sample complexity bounds for MoR-MAX are better than R-MAX in terms of $|S|$, $R_{\max}$, and $1/\epsilon$, and the same in terms of $|A|$ and $1/(1-\gamma)$.

Model-based Interval Estimation (MBIE) [16] improves on the empirical performance of R-MAX by computing confidence intervals on the reward and transition functions.[4] Like R-MAX, MBIE initialises all $\tilde{Q}$-values to $V_{\max}$. Thereafter, at each timestep, it computes the $\tilde{Q}$-values using VI with the following equation in place of the Bellman optimality equation:

$$\tilde{Q}(s,a) \leftarrow \hat{R}(s,a)+\gamma\sum_{s'\in S}\hat{T}(s,a,s')\max_{a'\in A}\tilde{Q}(s',a')+\frac{\beta}{\sqrt{n(s,a)}},$$

where β is an input parameter controlling the balance between exploration and exploitation. Thus, MBIE provides an exploration bonus that drives the agent towards state-action pairs that have been visited fewer times. With probability $1-\delta$, the sample complexity of MBIE is:

$$O\left(\frac{|S||A|R_{\max}^3}{\epsilon^3(1-\gamma)^6}\Big(|S|+\ln\frac{|S||A|R_{\max}}{\delta\epsilon(1-\gamma)}\Big)\ln\frac{1}{\delta}\ln\frac{R_{\max}}{\epsilon(1-\gamma)}\right).$$

The bounds for MBIE are similar to those of R-MAX, except in log factors, where they are worse in terms of $R_{\max}$, $1/\epsilon$, and $1/(1-\gamma)$.

3. METHOD

A critical weakness of R-MAX is that it performs poorly early in learning: since state-action pairs are indistinguishable until they have been experienced m times, R-MAX can only explore randomly during this phase. In essence, this poor performance is due to excessive optimism. To illustrate this point, consider the following example: an MDP with a single state s and two actions a_1 and a_2, which always yield rewards of 0 and $R_{\max}$, respectively. Suppose that at some timestep t, both (s,a_1) and (s,a_2) have been experienced n times each, where $n < m$.[5] Since R-MAX is still uncertain about both state-action pairs, it assumes that $\tilde{R}_t(s,a_1) = \tilde{R}_t(s,a_2) = R_{\max}$ and thus chooses an action randomly. However, given that the n experiences of (s,a_1) produced a reward of 0, R-MAX is clearly overly optimistic about $R(s,a_1)$. Even if a_1 always generates a reward of $R_{\max}$ in the future, $\hat{R}(s,a_1)$ will be at most $(0*n+(m-n)*R_{\max})/m = (m-n)R_{\max}/m$ by the end of learning, while $\hat{R}(s,a_2)$ can obviously be as high as $R_{\max}$.

The key idea behind V-MAX, our novel PAC-MDP algorithm, is to exploit this insight to temper the excessive optimism of R-MAX. Rather than assuming all state-action pairs visited fewer than m times have maximal value, V-MAX assumes only that the remaining $m-n$ visits will yield maximal return. Hence, V-MAX remains optimistic but can more quickly exploit what it learns. In our example, V-MAX would only need to choose a_1 once to know it is sub-optimal, thereby improving performance early in learning. In the remainder of this section, we formalise the V-MAX algorithm and prove bounds on its sample complexity.

3.1 The V-MAX Algorithm

V-MAX initialises $\tilde{Q}$, $\tilde{R}$, and $\tilde{T}$ exactly as R-MAX does. Unlike R-MAX, however, V-MAX updates $\tilde{R}$ and $\tilde{T}$ at each timestep, using the following equations:

$$\tilde{R}(s,a) = \frac{n(s,a)\hat{R}(s,a)+(m-n(s,a))R_{\max}}{m} \quad (2)$$

$$\tilde{T}(s,a,s') = \begin{cases}\dfrac{n(s,a)\hat{T}(s,a,s')}{m} & \text{if } s' \neq z\\ \dfrac{m-n(s,a)}{m} & \text{if } s' = z.\end{cases} \quad (3)$$

Thus, both the reward and transition function updates mix the observed rewards and transitions with the optimistic assumption that future samples will involve transitions to the maximally rewarding state z. V-MAX also updates $\tilde{Q}$ at each timestep, using VI on the MDP $(S\cup\{z\}, A, \tilde{R}, \tilde{T}, \gamma)$, and then simply follows a greedy policy with respect to $\tilde{Q}$.

A simpler alternative implementation can be derived that uses the empirical reward and transition functions $\hat{R}$ and $\hat{T}$ to compute $\tilde{Q}$ directly, thus eliminating the need to compute $\tilde{R}$ and $\tilde{T}$ and explicitly represent z. Substituting Equations 2 and 3 into the Bellman optimality equation (and omitting (s,a) in the notation for brevity), yields:

$$\begin{aligned}\tilde{Q} &= \frac{n\hat{R}+(m-n)R_{\max}}{m}+\gamma\Big(\frac{m-n}{m}\max_{a'\in A}\tilde{Q}(z,a')\\ &\quad+\sum_{s'\in S}\frac{n\hat{T}(s')}{m}\max_{a'\in A}\tilde{Q}(s',a')\Big)\\ &= \frac{n}{m}\Big(\hat{R}+\gamma\sum_{s'\in S}\hat{T}(s')\max_{a'\in A}\tilde{Q}(s',a')\Big)\\ &\quad+\Big(\frac{m-n}{m}\Big)\Big(R_{\max}+\gamma\max_{a'\in A}\tilde{Q}(z,a')\Big)\\ &= \frac{n}{m}\Big(\hat{R}+\gamma\sum_{s'\in S}\hat{T}(s')\max_{a'\in A}\tilde{Q}(s',a')\Big)+\Big(1-\frac{n}{m}\Big)V_{\max}.\end{aligned}$$

[3] There are minor differences in the bounds we state, since [13] assume that $R_{\max}=1$. Also, their bounds include the use of an admissible heuristic for initialising Q-values.

[4] We describe a variant of MBIE that Strehl and Littman call *MBIE with exploration bonus*. We consider only this variant because it is simpler and refer to it as MBIE for conciseness.

[5] We assume the agent does not know that the MDP is deterministic, and thus $m > 1$.

The resulting implementation of V-MAX, described in Algorithm 2, initialises $\tilde{Q}(s,a)$ to $V_{\max}$ for all $(s,a) \in S \times A$. At each timestep, it updates $\tilde{Q}$ using VI on the MDP $(S, A, \hat{R}, \hat{T}, \gamma)$ but with the following equation in place of the Bellman optimality equation:

$$\tilde{Q}(s,a) = \frac{n(s,a)}{m}\left(\hat{R}(s,a) + \gamma \sum_{s' \in S} \hat{T}(s,a,s') \max_{a' \in A} \tilde{Q}(s',a')\right) + \left(1 - \frac{n(s,a)}{m}\right) V_{\max}. \tag{4}$$

Finally, the agent follows a greedy policy with respect to $\tilde{Q}$. The name V-MAX is inspired by this implementation, as it computes an optimistic value function directly from $V_{\max}$.

Algorithm 2: V-MAX

Input: S, A, γ, m, ϵ_1, $R_{\max}$

1 $V_{\max} \leftarrow R_{\max}/(1-\gamma)$
2 **foreach** $(s,a) \in S \times A$ **do**
3 $n(s,a) \leftarrow 0$
4 $r(s,a) \leftarrow 0$
5 **foreach** $s' \in S$ **do** $n(s,a,s') \leftarrow 0$
6 $\tilde{Q}(s,a) \leftarrow V_{\max}$
7 **end**
8 **for** $t = 1, 2, 3, \ldots$ **do**
9 Observe current state s
10 Execute action $a := \text{argmax}_{a' \in A} \tilde{Q}(s,a')$
11 Observe immediate reward r and next state s'
12 **if** $n(s,a) < m$ **then**
13 $n(s,a) \leftarrow n(s,a) + 1$
14 $r(s,a) \leftarrow r(s,a) + r$
15 $n(s,a,s') \leftarrow n(s,a,s') + 1$
16 $\hat{R}(s,a) \leftarrow r(s,a)/n(s,a)$
17 **foreach** $s' \in S$ **do** $\hat{T}(s,a,s') \leftarrow n(s,a,s')/n(s,a)$
18 **repeat**
19 $\Delta \leftarrow 0$
20 **foreach** $(s,a) \in S \times A$ **do**
21 **if** $n(s,a) > 0$ **then**
22 $q \leftarrow \tilde{Q}(s,a)$
23 $Q \leftarrow \hat{R}(s,a) + \gamma \sum_{s' \in S} \hat{T}(s,a,s') \max_{a' \in A} \tilde{Q}(s',a')$
24 $\tilde{Q}(s,a) \leftarrow (n(s,a)/m)Q + (1 - n(s,a)/m)V_{\max}$
25 $\Delta \leftarrow \max(\Delta, |q - \tilde{Q}(s,a)|)$
26 **end**
27 **end**
28 **until** $\Delta \leq \epsilon_1$
29 **end**
30 **end**

3.2 Sample Complexity of V-MAX

Section 4 will demonstrate that V-MAX can substantially outperform performance R-MAX. Here, we show that these empirical advantages do not come at the expense of its theoretical properties, by proving upper bounds on its sample complexity identical to those of R-MAX. As the example in Section 3 illustrates, V-MAX can explore less than R-MAX. Thus, the bounds we prove here are critical for ensuring that V-MAX's tempered optimism does not increase the chance of converging to a suboptimal model.

We begin with supporting lemmas showing that the value function used by V-MAX decreases monotonically and is always at most $V_{\max}$. We then prove the main result in Theorem 1, using techniques similar to [13] and [16].

In the following, let the *value iteration step* (vstep) i be the i^{th} value iteration update (Lines 23-24 of Algorithm 2) since $t = 1$. Let $n_{[i]}$, $\tilde{Q}_{[i]}$, $\hat{R}_{[i]}$, and $\hat{T}_{[i]}$ denote the values of n, $\tilde{Q}$, $\hat{R}$, and $\hat{T}$, respectively, at vstep i. Finally, let $(s_{[i]}, a_{[i]})$ be the state-action pair whose $\tilde{Q}$-value is updated at vstep i. As with other PAC-MDP algorithms, we assume that all rewards are non-negative, and are upper bounded by $R_{\max}$.

LEMMA 1. *For all $(s,a) \in S \times A$, and for all policies π and timesteps t:*

$$\tilde{Q}^{\pi}_{t}(s,a) \leq V_{\max}.$$

PROOF. Using weak induction on the vsteps i, we prove that for all $(s,a) \in S \times A$ and for all policies π, $\tilde{Q}^{\pi}_{[i]}(s,a) \leq V_{\max}$. The base case, that $\tilde{Q}^{\pi}_{[0]}(s,a) = V_{\max}$ for all $(s,a) \in S \times A$, holds because all $\tilde{Q}$-values are initialised to $V_{\max}$. In the inductive step, we assume that, for all $(s,a) \in S \times A$, $\tilde{Q}^{\pi}_{[k]}(s,a) \leq V_{\max}$ and must prove that $\tilde{Q}^{\pi}_{[k+1]}(s,a) \leq V_{\max}$.

Note that $(s_{[k+1]}, a_{[k+1]})$ is the only state-action pair whose value is updated at vstep $k+1$, so $\tilde{Q}^{\pi}_{[k+1]}(s',a') = \tilde{Q}^{\pi}_{[k]}(s',a')$ for all other $(s',a') \neq (s_{[k+1]}, a_{[k+1]})$. Thus, by the inductive hypothesis, $\tilde{Q}^{\pi}_{[k+1]}(s',a') \leq V_{\max}$. Consequently, we only need to prove that:

$$\tilde{Q}^{\pi}_{[k+1]}(s_{[k+1]}, a_{[k+1]}) \leq V_{\max}$$

This can be derived from Equation 4 as follows (omitting $(s_{[k+1]}, a_{[k+1]})$ in the notation for brevity).

$$\begin{aligned}
\tilde{Q}^{\pi}_{[k+1]} &= \frac{n_{[k+1]}}{m}\left(\hat{R}_{[k+1]} + \gamma \sum_{s' \in S} \hat{T}_{[k+1]}(s') \tilde{Q}^{\pi}_{[k]}(s', \pi(s'))\right) \\
&\quad + \left(1 - \frac{n_{[k+1]}}{m}\right) V_{\max} \\
&\leq \frac{n_{[k+1]}}{m}\left(R_{\max} + \gamma \sum_{s' \in S} \hat{T}_{[k+1]}(s') V_{\max}\right) \\
&\quad + \left(1 - \frac{n_{[k+1]}}{m}\right) V_{\max} \\
&\leq \frac{n_{[k+1]}}{m} V_{\max} + \left(1 - \frac{n_{[k+1]}}{m}\right) V_{\max} \\
&\leq V_{\max} \quad \square
\end{aligned}$$

LEMMA 2. *For all $(s,a) \in S \times A$, and for all policies π and timesteps $t > 0$:*

$$\tilde{Q}^{\pi}_{t}(s,a) \leq \tilde{Q}^{\pi}_{t-1}(s,a).$$

PROOF. Using strong induction on the vsteps i, we prove that, for all $(s,a) \in S \times A$, $\tilde{Q}^{\pi}_{[i]}(s,a) \leq \tilde{Q}^{\pi}_{[i-1]}(s,a)$, i.e., $\tilde{Q}^{\pi}(s,a)$ never increases as the number of value iteration updates increases. For the base case, $\tilde{Q}^{\pi}_{[0]}(s,a) = V_{\max}$ for all $(s,a) \in S \times A$ since all $\tilde{Q}$-values are initialised to $V_{\max}$. Also, from Lemma 1, $\tilde{Q}^{\pi}_{[1]}(s,a) \leq V_{\max}$. Thus $\tilde{Q}^{\pi}_{[1]}(s,a) \leq \tilde{Q}^{\pi}_{[0]}(s,a)$. For the inductive step, we assume that, for all $(s,a) \in S \times A$, and for all j such that $0 < j \leq k$, $\tilde{Q}^{\pi}_{[j]}(s,a) \leq \tilde{Q}^{\pi}_{[j-1]}(s,a)$. We need to prove that $\tilde{Q}^{\pi}_{[k+1]}(s,a) \leq \tilde{Q}^{\pi}_{[k]}(s,a)$.

Note that $(s_{[k+1]}, a_{[k+1]})$ is the only state-action pair whose value is updated at vstep $k+1$, so $\tilde{Q}^{\pi}_{[k+1]}(s',a') = \tilde{Q}^{\pi}_{[k]}(s',a')$ for all other $(s',a') \neq (s_{[k+1]}, a_{[k+1]})$. Now, let τ be the vstep prior to $k+1$ at which $\tilde{Q}(s_{[k+1]}, a_{[k+1]})$ was last updated.[6]

[6]If $\tilde{Q}(s_{[k+1]}, a_{[k+1]})$ was never updated, then $\tilde{Q}^{\pi}_{[k+1]}(s_{[k+1]}, a_{[k+1]}) = \tilde{Q}^{\pi}_{[k]}(s_{[k+1]}, a_{[k+1]})$.

Since $\tau \leq k$, and $\tilde{Q}^{\pi}(s_{[k+1]}, a_{[k+1]})$ is unchanged between vsteps τ and $k+1$, we need only prove that:

$$\tilde{Q}^{\pi}_{[k+1]}(s_{[k+1]}, a_{[k+1]}) \leq \tilde{Q}^{\pi}_{[\tau]}(s_{[k+1]}, a_{[k+1]}).$$

Let c_n be the number of times $n(s_{[k+1]}, a_{[k+1]})$ is updated between vsteps τ and $k+1$.[7] For the c_n updates, let c_R be the total (undiscounted) reward accumulated and $c_T(s')$ be the number of times that s' is the next state observed. Again omitting $(s_{[k+1]}, a_{[k+1]})$, we can write:

$$n_{[k+1]} = n_{[\tau]} + c_n \tag{5}$$

$$\hat{R}_{[k+1]} = \frac{n_{[\tau]}\hat{R}_{[\tau]} + c_R}{n_{[\tau]} + c_n} \tag{6}$$

$$\hat{T}_{[k+1]}(s') = \frac{n_{[\tau]}\hat{T}_{[\tau]}(s') + c_T(s')}{n_{[\tau]} + c_n} \text{ for all } s' \in S. \tag{7}$$

Letting $\tilde{V}^{\pi}_{[i]}(s')$ denote $\tilde{Q}^{\pi}_{[i]}(s', \pi(s'))$, we use Equations 5–7 to reduce Equation 4:

$$\begin{aligned}
\tilde{Q}^{\pi}_{[k+1]} &= \left(1 - \frac{n_{[k+1]}}{m}\right)V_{\max} + \frac{n_{[k+1]}}{m}\left(\hat{R}_{[k+1]}\right. \\
&\qquad \left. + \gamma \sum_{s' \in S} \hat{T}_{[k+1]}(s')\tilde{V}^{\pi}_{[k]}(s')\right) \\
&= \left(1 - \frac{n_{[\tau]} + c_n}{m}\right)V_{\max} + \frac{n_{[\tau]} + c_n}{m}\left(\frac{n_{[\tau]}\hat{R}_{[\tau]} + c_R}{n_{[\tau]} + c_n}\right. \\
&\quad \left. + \gamma \sum_{s' \in S} \frac{n_{[\tau]}\hat{T}_{[\tau]}(s') + c_T(s')}{n_{[\tau]} + c_n}\tilde{V}^{\pi}_{[k]}(s')\right) \\
&= \left(1 - \frac{n_{[\tau]}}{m} - \frac{c_n}{m}\right)V_{\max} + \frac{1}{m}\left(n_{[\tau]}\hat{R}_{[\tau]} + c_R\right. \\
&\quad \left. + \gamma \sum_{s' \in S} \left(n_{[\tau]}\hat{T}_{[\tau]}(s') + c_T(s')\right)\tilde{V}^{\pi}_{[k]}(s')\right) \\
&= \frac{n_{[\tau]}}{m}\left(\hat{R}_{[\tau]} + \gamma \sum_{s' \in S} \hat{T}_{[\tau]}(s')\tilde{V}^{\pi}_{[k]}(s')\right) + \left(1 - \frac{n_{[\tau]}}{m}\right)V_{\max} \\
&\quad + \frac{1}{m}\left(c_R - c_n V_{\max} + \gamma \sum_{s' \in S} c_T(s')\tilde{V}^{\pi}_{[k]}(s')\right).
\end{aligned}$$

We know that $c_R \leq c_n R_{\max}$ and $\sum_{s' \in S} c_T(s') = c_n$. Also, since $\tau - 1 < k$, the inductive hypothesis implies that for all $s' \in S$, $\tilde{V}^{\pi}_{[k]}(s') \leq \tilde{V}^{\pi}_{[\tau-1]}(s')$. Lastly, Lemma 1 implies that $\tilde{V}^{\pi}_{[k]}(s') \leq V_{\max}$ for all $s' \in S$. We use these facts to further reduce $\tilde{Q}^{\pi}_{[k+1]}$:

$$\begin{aligned}
&\leq \frac{n_{[\tau]}}{m}\left(\hat{R}_{[\tau]} + \gamma \sum_{s' \in S} \hat{T}_{[\tau]}(s')\tilde{V}^{\pi}_{[\tau-1]}(s')\right) + \left(1 - \frac{n_{[\tau]}}{m}\right)V_{\max} \\
&\quad + \frac{1}{m}\left(c_n R_{\max} - c_n V_{\max} + \gamma \sum_{s' \in S} c_T(s')V_{\max}\right) \\
&\leq \tilde{Q}^{\pi}_{[\tau]} + \frac{1}{m}\left(c_n R_{\max} - c_n V_{\max} + \gamma \sum_{s' \in S} c_T(s')V_{\max}\right) \\
&\leq \tilde{Q}^{\pi}_{[\tau]} \quad \square
\end{aligned}$$

[7] While $c_n \in \{0, 1\}$ for V-MAX, it could be larger if full VI was not performed at each timestep.

THEOREM 1. *Suppose that $0 < \epsilon < V_{\max}$ and $0 < \delta < 1$ are two real numbers and $M = (S, A, R, T, \gamma)$ is any MDP. There exist $m = O\left(\frac{(|S|+\ln(|S||A|/\delta))R^2_{\max}}{\epsilon^2(1-\gamma)^4}\right)$ and $\epsilon_1 = O(\epsilon)$ such that if V-MAX is executed on M with inputs m and ϵ_1, then the following holds. Let $\mathcal{A}_t$ denote V-MAX's policy at time t, and let s_t denote the state at time t. With probability at least $1 - \delta$, $V^{\mathcal{A}_t}_M(s_t) \geq V^*_M(s_t) - \epsilon$ is true for all but*

$$O\left(\frac{|S||A|R^3_{\max}}{\epsilon^3(1-\gamma)^6}\left(|S| + \ln\frac{|S||A|}{\delta}\right)\ln\frac{1}{\delta}\ln\frac{R_{\max}}{\epsilon(1-\gamma)}\right)$$

timesteps t.

PROOF. Let $m = O\left(\frac{(|S|+\ln(|S||A|/\delta))R^2_{\max}}{\epsilon_1^2(1-\gamma)^4}\right)$ and let K_t be the set of state-action pairs experienced at least m times by timestep t. Let $H = \lceil \frac{1}{1-\gamma}\ln\frac{R_{\max}}{\epsilon_1(1-\gamma)} \rceil$ and E_M be the event that a state-action pair not in K_t is encountered in a trial generated by starting from s_t and following $\mathcal{A}_t$ for H timesteps in M. We consider two mutually exclusive cases.

In the first case, $\Pr(E_M) \geq \epsilon_1/V_{\max}$. We treat H timesteps in M as one toss of a weighted coin, and E_M as the event that it shows heads. If we see $m|S||A|$ heads, then all $(s, a) \in S \times A$ are in K. As a result of the Chernoff-Hoeffding bound, with probability $1-\delta$, after j tosses, $m|S||A|$ or more heads are seen, for some $j = O(\frac{m|S||A|V_{\max}}{\epsilon_1}\ln\frac{1}{\delta})$. Since each toss is equivalent to H timesteps in M, with probability $1-\delta$, all state-action pairs are in K after $O(\frac{m|S||A|HR_{\max}}{\epsilon_1(1-\gamma)}\ln\frac{1}{\delta})$ timesteps. Also, $\Pr(E_M) = 0$ once all state-action pairs are in K. Thus, given m and H, $\Pr(E_M) < \epsilon_1/V_{\max}$ after

$$O\left(\frac{|S||A|R^3_{\max}}{\epsilon_1^3(1-\gamma)^6}\left(|S| + \ln\frac{|S||A|}{\delta}\right)\ln\frac{1}{\delta}\ln\frac{R_{\max}}{\epsilon_1(1-\gamma)}\right)$$

timesteps.

In the second case, $\Pr(E_M) < \epsilon_1/V_{\max}$. Recall that for some policy $\mathcal{A}_t$ on the MDP M, $V^{\mathcal{A}_t}_M$ denotes the true value function. Let $V(s, T)$ denote the T-step value of s. Given H and Lemma 2 of [9]:

$$V^{\mathcal{A}_t}_M(s_t) \geq V^{\mathcal{A}_t}_M(s_t, H).$$

Let z be a fictitious state, and let $\bar{S} = S \cup \{z\}$. Let $\bar{M} = (\bar{S}, A, \bar{R}, \bar{T}, \gamma)$ be an MDP, where for all $(s, a, s') \in S \times A \times S$, $\bar{R}(s, a) = R(s, a)$, $\bar{T}(s, a, s') = T(s, a, s')$, and $\bar{T}(s, a, z) = 0$. $\bar{R}(z, .)$ and $\bar{T}(z, ., .)$ are set arbitrarily. Since $s_t \neq z$ and the reward and transition functions in M and $\bar{M}$ are equal on $S \times A \times S$, $V^{\mathcal{A}_t}_M(s_t, H) = V^{\mathcal{A}_t}_{\bar{M}}(s_t, H)$. Thus:

$$V^{\mathcal{A}_t}_M(s_t) \geq V^{\mathcal{A}_t}_{\bar{M}}(s_t, H).$$

Let $M_K = (\bar{S}, A, R_K, T_K, \gamma)$ be an MDP with $R_K = \bar{R}$ and $T_K(.) = \bar{T}(.)$ for state-action pairs in K_t, while $R_K = \tilde{R}_t$ and $T_K(.) = \tilde{T}_t(.)$ for state-action pairs not in K_t, where $\tilde{R}_t$ and $\tilde{T}_t$ are defined with Equations 2 and 3. Let $E_{\bar{M}}$ be the event that a state-action pair not in K_t is encountered in a trial generated by starting from s_t and following $\mathcal{A}_t$ for H timesteps in $\bar{M}$. Lemma 8 of [15] implies that $V^{\mathcal{A}_t}_{\bar{M}}(s_t, H) \geq V^{\mathcal{A}_t}_{M_K}(s_t, H) - V_{\max}\Pr(E_{\bar{M}})$. However, since M and $\bar{M}$ differ only on z, which is never encountered, $\Pr(E_{\bar{M}}) = \Pr(E_M) < \epsilon_1/V_{\max}$. It follows that:

$$V^{\mathcal{A}_t}_M(s_t) \geq V^{\mathcal{A}_t}_{M_K}(s_t, H) - \epsilon_1.$$

From Lemma 2 of [9], $V^{\mathcal{A}_t}_{M_K}(s_t, H) \geq V^{\mathcal{A}_t}_{M_K}(s_t) - \epsilon_1$. Thus:

$$V^{\mathcal{A}_t}_M(s_t) \geq V^{\mathcal{A}_t}_{M_K}(s_t) - 2\epsilon_1.$$

Let $\hat{M}_K = (\bar{S}, A, \hat{R}_K, \hat{T}_K, \gamma)$ be an MDP with $\hat{R}_K = \hat{R}_t$ and $\hat{T}_K(.) = \hat{T}_t(.)$ for state-action pairs in K_t, while $\hat{R}_K = \tilde{R}_t$ and $\hat{T}_K(.) = \tilde{T}_t(.)$ for state-action pairs not in K_t. Note that $\hat{M}_K$ and M_K have equal reward and transition functions for all state-action pairs not in K_t. Thus, given m, Lemma 15 of [13] implies[8] that with probability at least $1-\delta$, $V_{M_K}^{\mathcal{A}_t}(s) \geq V_{\hat{M}_K}^{\mathcal{A}_t}(s) - \epsilon_1$ for all $s \in S$. Consequently, with probability at least $1-\delta$:

$$V_M^{\mathcal{A}_t}(s_t) \geq V_{\hat{M}_K}^{\mathcal{A}_t}(s_t) - 3\epsilon_1.$$

Now, let $\hat{M}_t = (S, A, \hat{R}_t, \hat{T}_t, \gamma)$ be the MDP that is learned by V-MAX using maximum likelihood estimation. We know that $\hat{R}_t = \tilde{R}_K$ and $\hat{T}_t(.) = \tilde{T}_K(.)$ for state-action pairs not in K_t. For these pairs, the derivation in Section 3.1 implies that computing $\tilde{Q}$-values using Equation 4 for VI in $\hat{M}_t$ is equivalent to computing Q-values using the Bellman optimality equation for VI in $\hat{M}_K$. Furthermore, for state-action pairs in K_t, Equation 4 reduces to the Bellman optimality equation. Thus, for all $s \in S$, $V_{\hat{M}_K}^{\mathcal{A}_t}(s) = \tilde{V}_{\hat{M}_t}^{\mathcal{A}_t}(s)$. Hence, with probability at least $1-\delta$:

$$V_M^{\mathcal{A}_t}(s_t) \geq \tilde{V}_{\hat{M}_t}^{\mathcal{A}_t}(s_t) - 3\epsilon_1.$$

Since $\mathcal{A}_t$ is greedy with respect to $\tilde{Q}_{\hat{M}_t}$, for all $s \in S$, and all policies π, $\tilde{V}_{\hat{M}_t}^{\mathcal{A}_t}(s) \geq \tilde{V}_{\hat{M}_t}^{\pi}(s)$. Let π^* denote the optimal policy for M. Thus, with probability at least $1-\delta$:

$$V_M^{\mathcal{A}_t}(s_t) \geq \tilde{V}_{\hat{M}_t}^{\pi^*}(s_t) - 3\epsilon_1.$$

Let $\hat{M}_x = (S, A, \hat{R}_x, \hat{T}_x, \gamma)$ be an MDP where $\hat{R}_x$ and $\hat{T}_x$ are maximum likelihood estimates of R and T at some timestep $x \geq t$, such that all $(s,a) \in S \times A$ are in K_x. Then, Lemma 2 implies that for all $s \in S$ and all policies π, $\tilde{V}_{\hat{M}_t}^{\pi}(s) \geq \tilde{V}_{\hat{M}_x}^{\pi}(s)$. Equation 4 reduces to the Bellman optimality equation for all $(s,a) \in K_x$, implying that for all $s \in S$, and for all policies π, $\tilde{V}_{\hat{M}_x}^{\pi}(s) = V_{\hat{M}_x}^{\pi}(s)$. Consequently, with probability at least $1-\delta$:

$$V_M^{\mathcal{A}_t}(s_t) \geq V_{\hat{M}_x}^{*}(s_t) - 3\epsilon_1.$$

Given our choice of m, an application of Lemma 15 of [13] yields that with probability at least $1-\delta$, $V_{\hat{M}_x}^{*}(s) \geq V_M^{*}(s) - \epsilon_1$, for all $s \in S$. Thus, with probability at least $1-\delta$:

$$V_M^{\mathcal{A}_t}(s_t) \geq V_M^{*}(s_t) - 4\epsilon_1.$$

Setting $\epsilon_1 = \epsilon/4$ yields the desired result. □

4. EXPERIMENTS

We evaluate the empirical performance of V-MAX on two tasks. As comparisons, we use R-MAX and MoR-MAX, because of their close relationship to V-MAX, and MBIE, because of its strong empirical track record: no other PAC-MDP method has been shown to substantially outperform it. On the contrary, it has greatly outperformed E^3 [9], R-MAX, and Delayed Q-Learning [13] and matched the performance of Optimistic Initial Model (OIM) [18].[9]

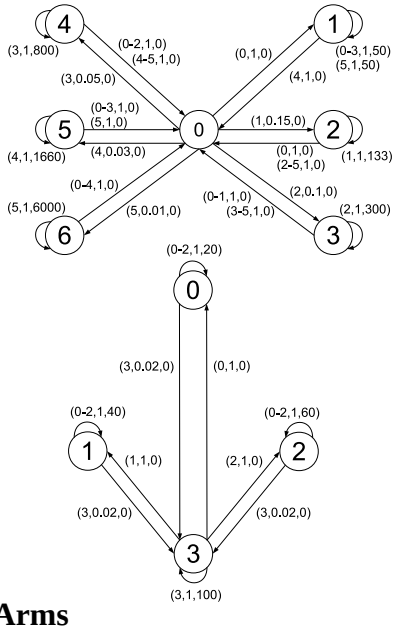
Since our goal is to show that V-MAX advances the PAC-MDP state of the art, we restrict our analysis to PAC-MDP algorithms. Comparisons to efficient-exploration algorithms developed under other frameworks, such as *regret* [5] and *knows what it knows* (KWIK) [11], are left to future work.

Following [14] and [16], we use cumulative (undiscounted) reward as our evaluation metric. Even though maximising the cumulative reward is not the goal of PAC-MDP methods, it is a suitable metric for our experiments because it shows that V-MAX can accrue more reward while matching the sample complexity of R-MAX.

For each algorithm, performance is affected by the value of its critical parameters: ϵ_1 and m for R-MAX, MoR-MAX, and V-MAX and ϵ_1 and C (where $\beta = CV_{\max}$) for MBIE. For all algorithms, we set ϵ_1 to 0.01, at which VI finds the optimal policy. For each MDP, we first measure the average cumulative reward after 25,000 timesteps over 100 runs on each MDP across a large range of values for m and C. Using the results, we select a smaller range of parameter values likely to perform well. For our final results, we measure the average cumulative reward over 1000 runs across this selected range. For more timesteps, our results are qualitatively similar. For significantly fewer timesteps, none of the algorithms learn the optimal policy, thus biasing the results.

For each algorithm, if two or more actions have the same value, the agent takes the action that has been tried fewer times. Remaining ties are broken by choosing actions in ascending order, e.g., action 0 is chosen before action 1. This action-selection method is a simple way to ensure all algorithms try lower-valued actions before discovering the optimal action in the highest-valued state, without giving any algorithm an advantage.

Figure 1: SixArms and Anchor MDPs. Each vertex denotes a state and each edge a state transition. A tuple (a, p, r) indicates that action a causes the given transition with probability p and reward r and causes a self-transition with probability $1-p$ and reward 0.



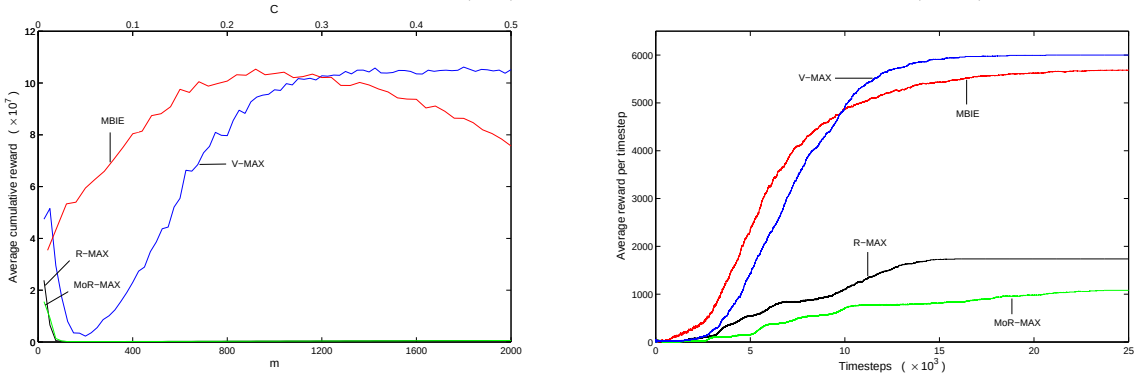
4.1 SixArms

The SixArms MDP in Figure 1 (top) was used to show that MBIE can outperform R-MAX, which in turn outperforms both E^3 and ϵ-greedy exploration [14, 16]. Thus, it is an important benchmark test for V-MAX. The agent starts in state 0 and chooses between 6 actions, each of which may

[8]While lemma 15 of [13] is stated for R-MAX, the same result holds for V-MAX, from an application of Lemmas 13 and 14 of [13] in lemma 2 of [16].

[9]OIM also has substantially weaker sample complexity bounds than R-MAX, V-MAX, and MBIE.

Figure 2: Average cumulative reward (left) and average reward per timestep (right) on SixArms.



lead to states 1-6. Once there, the agent can remain and receive a reward at each timestep or return to state 0. The optimal policy is to try to reach state 6 and then repeatedly choose action 5. Since the probability of reaching state 6 is small, the agent typically must try each action several times to discover the optimal policy. The discount factor is 0.95.

Figure 2 (left) shows the average cumulative reward over a range of parameter values for each algorithm (C for MBIE, and m for the other algorithms). V-MAX vastly outperforms R-MAX and MoR-MAX and matches the performance of MBIE. In addition, Figure 2 (right), which shows the average reward per timestep for each algorithm at optimal parameter values, demonstrates that V-MAX is the only algorithm able to consistently achieve the maximum reward per timestep. MBIE obtains similar cumulative reward by gaining higher rewards in the initial phases of the experiment.

Note that MoR-MAX, which has the strongest sample complexity bounds, has the worst empirical performance. To see why, consider state 0. A problematic scenario for R-MAX, MoR-MAX, and V-MAX is where the agent never transitions from state 0 to any of the states 2 and above. Then, it will forever assume that, in state 0, all actions other than 0 cause a self-transition. However, to avoid this scenario, R-MAX and V-MAX need only ensure with high probability that this does not happen in the *first* m samples for each state-action pair. By contrast, MoR-MAX repeatedly collects batches of m samples per state-action pair and discards them when new samples lead to a lower value for that pair. Consequently, MoR-MAX must ensure with high probability that this scenario does not occur on *any* batch of m samples. In general, this requires higher values of m, yielding too much initial exploration.

Overall, results on SixArms show that V-MAX can match the empirical performance of MBIE while maintaining the stronger theoretical guarantees of R-MAX. Furthermore, Figure 2 (left) shows that, as long as m is set conservatively, V-MAX's performance is invariant to it, essentially eliminating the need for parameter tuning. In contrast, MBIE requires careful tuning of C to match V-MAX's performance.

4.2 Anchor

In SixArms, MBIE accrues more reward early in learning by more often choosing relatively good, but suboptimal, actions in states 1-5. However, in this MDP, exploiting such local optima has little cost, since the agent can at any time deterministically return to state 0. Therefore, we hypothesize that MBIE will perform poorly in MDPs where choosing locally optimal actions significantly impedes the agent's progress towards better state-action pairs.

To test this hypothesis, we use the Anchor MDP shown in Figure 1 (bottom). The agent starts in state 0 and can choose, in states 0-2, to remain and receive a sub-optimal reward or try to reach state 3, where maximal reward is available. Since the agent initially chooses actions in ascending order, it must try each action in each states 0-2 at least once before discovering that action 3 in state 3 yields reward $R_{\max} = 100$. The discount factor is 0.99.

The optimal policy is to choose action 3 in each state. However, in each state 0-2, action 3 is the only one that produces no reward. Furthermore, all actions in these states usually produce self-transitions, with only action 3 transitioning to state 3 with low probability. Consequently, to determine the optimal policy efficiently, an agent must avoid being distracted by the sub-optimal rewards offered by actions 0-2 and continue to attempt action 3. Unlike in SixArms, the agent cannot deterministically escape the local optima (in states 0-2), making efficient exploration crucial.

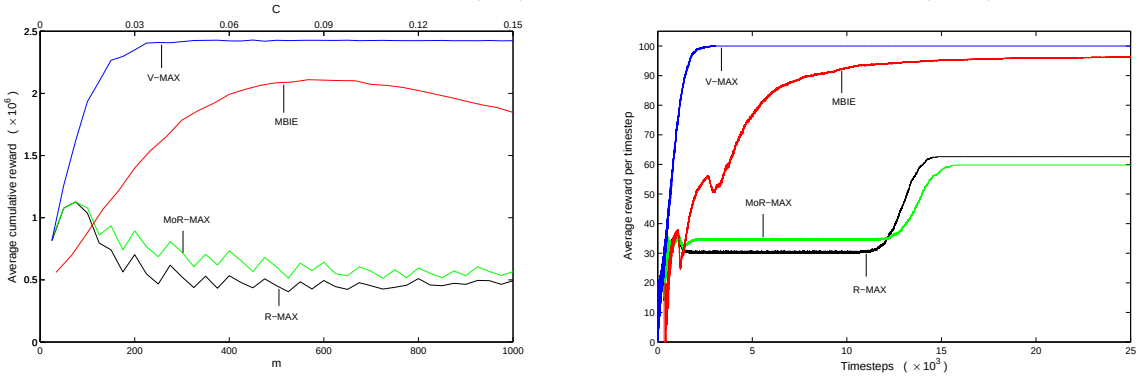
Figure 3 (left) shows the average cumulative reward for each algorithm on Anchor. As in SixArms, V-MAX substantially outperforms both R-MAX and MoR-MAX. However, it also substantially outperforms MBIE, supporting the hypothesis that MBIE is vulnerable to distraction by local optima. In addition, as in SixArms, the performance of V-MAX is invariant for large choices of m, whereas MBIE again requires careful tuning of C.

In Figure 3 (left), the performance of both R-MAX and MoR-MAX oscillates as m changes. We suspect this is a result of R-MAX and MoR-MAX periodically cycling between the states 0 to 2 during exploration. Since m affects how many timesteps they spend in each of these states, performance can increase for values of m in which time expires just before the agent cycles to the low reward states 0 and/or 1. However, since increasing m increases exploration, the performance of R-MAX and MoR-MAX decreases overall, despite the oscillations, as m is increased beyond 100.

Figure 3 (right) shows the average reward per timestep in Anchor. Only V-MAX consistently accrues maximal reward per timestep. Although MBIE gets close, it takes significantly longer than V-MAX to do so. In addition, both R-MAX and MoR-MAX remain in states 1 and 2 for a long time, constantly receiving the sub-optimal rewards of 40 and 60, respectively. Though the effect is less extreme for MBIE, its performance also dips in the same places.

Overall, the SixArms and Anchor experiments demonstrate that 1) V-MAX can greatly outperform both R-MAX and MoR-MAX, 2) V-MAX can match MBIE and, on problems with multiple local optima, substantially outperform it, and 3) V-MAX, whose performance is invariant for large

Figure 3: Average cumulative reward (left) and average reward per timestep (right) on Anchor.



m, essentially obviates the need for parameter tuning while MBIE's performance is sensitive to the choice of C.

5. FUTURE WORK AND CONCLUSIONS

The development of V-MAX creates several opportunities for future work. We hope to extend our theoretical results by proving that, for the same m, V-MAX strictly dominates R-MAX on all MDPs in that it never makes an exploratory mistake that R-MAX does not make, given the same $\hat{R}$ and $\hat{T}$. In addition to sample complexity, per-timestep computational complexity may also be important in large MDPs. In such cases, techniques for reducing the computational cost of PAC-MDP algorithms [4, 12] may benefit V-MAX as well. Also, extensions to R-MAX that exploit problem structure, such as Fitted R-MAX [6], R-MAXQ [7] and RAM-Rmax [10], could be adapted to use V-MAX instead. Lastly, since we suspect that the core idea of tempered optimism is actually orthogonal to the quality of the bounds, it could be used to derive other PAC-MDP methods, e.g., MoV-MAX, with strong empirical performance and even tighter bounds.

Overall, the theoretical and empirical results presented in this paper demonstrate that V-MAX is a significant step forward for PAC RL, since it can outperform state-of-the-art PAC-MDP algorithms while maintaining attractive theoretical guarantees. In addition, these guarantees show for the first time that tempered optimism is possible without compromising sample complexity bounds. Finally, our experiments suggest that V-MAX requires little or no parameter tuning. Consequently, we hope that it will help make PAC-MDP algorithms more accessible for general usage.

6. ACKNOWLEDGMENTS

Thanks Lihong Li, Alexander Strehl, Michael Littman, and Harm van Seijen for their helpful comments.

7. REFERENCES

[1] R. E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.
[2] R. I. Brafman and M. Tennenholtz. R-MAX – a general polynomial time algorithm for near-optimal reinforcement learning. *Journal of Machine Learning Research*, 3:213–231, 2002.
[3] M. O. Duff. *Optimal Learning: Computational Procedures for Bayes-adaptive Markov Decision Processes*. PhD thesis, U. of Mass.-Amherst, 2002.
[4] M. Grzés and J. Hoey. Efficient planning in R-MAX. In *Proc. of AAMAS*, 2011.
[5] T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11:1563–1600, 2010.
[6] N. K. Jong and P. Stone. Model-based exploration in continuous state spaces. In *The Symposium on Abstraction, Reformulation, and Approximation*, 2007.
[7] N. K. Jong and P. Stone. Hierarchical model-based reinforcement learning: Rmax + MAXQ. In *Proc. of ICML*, 2008.
[8] S. M. Kakade. *On the Sample Complexity of Reinforcement Learning*. PhD thesis, University College London, 2003.
[9] M. J. Kearns and S. P. Singh. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.
[10] B. R. Leffler, M. L. Littman, and T. Edmunds. Efficient reinforcement learning with relocatable action models. In *Proc. of AAAI*, pages 572–577, 2007.
[11] L. Li, M. L. Littman, T. J. Walsh, and A. L. Strehl. Know what it knows: A framework for self-aware learning. *Machine Learning*, 82(3):399–443, 2011.
[12] A. L. Strehl, L. Li, and M. L. Littman. Incremental model-based learners with formal learning-time guarantees. In *Proc. of UAI*, pages 485–493, 2006.
[13] A. L. Strehl, L. Li, and M. L. Littman. Reinforcement learning in finite MDPs: PAC analysis. *Journal of Machine Learning Research*, 10:2413–2444, 2009.
[14] A. L. Strehl and M. L. Littman. An empirical evaluation of interval estimation for Markov decision processes. In *Proc. of ICTAI*, pages 129–135, 2004.
[15] A. L. Strehl and M. L. Littman. A theoretical analysis of model-based interval estimation. In *Proc. of ICML*, pages 857–864, 2005.
[16] A. L. Strehl and M. L. Littman. An analysis of model-based interval estimation for Markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008.
[17] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, Cambridge, MA, 1998.
[18] I. Szita and A. Lörincz. The many faces of optimism: A unifying approach. In *Proc. of ICML*, 2008.
[19] I. Szita and C. Szepesvári. Model-based reinforcement learning with nearly tight exploration complexity bounds. In *Proc. of ICML*, pages 1031–1038, 2010.
[20] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.
[21] C. J. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

Reinforcement Learning Transfer via Sparse Coding

Haitham B. Ammar
Maastricht University, The Netherlands
haitham.bouammar@maastrichtuniversity.nl

Karl Tuyls
Maastricht University, The Netherlands
k.tuyls@maastrichtuniversity.nl

Matthew E. Taylor
Lafayette College, USA
taylorm@lafayette.edu

Kurt Driessens
Maastricht University, The Netherlands
kurt.driessens@maastrichtuniversity.nl

Gerhard Weiss
Maastricht University, The Netherlands
gerhard.weiss@maastrichtuniversity.nl

ABSTRACT

Although reinforcement learning (RL) has been successfully deployed in a variety of tasks, learning speed remains a fundamental problem for applying RL in complex environments. Transfer learning aims to ameliorate this shortcoming by speeding up learning through the adaptation of previously learned behaviors in similar tasks. Transfer techniques often use an inter-task mapping, which determines how a pair of tasks are related. Instead of relying on a hand-coded inter-task mapping, this paper proposes a novel transfer learning method capable of autonomously creating an inter-task mapping by using a novel combination of sparse coding, sparse projection learning and sparse Gaussian processes. We also propose two new transfer algorithms (*TrLSPI* and *TrFQI*) based on least squares policy iteration and fitted-Q-iteration. Experiments not only show successful transfer of information between similar tasks, inverted pendulum to cart pole, but also between two very different domains: mountain car to cart pole. This paper empirically shows that the learned inter-task mapping can be successfully used to (1) improve the performance of a learned policy on a fixed number of environmental samples, (2) reduce the learning times needed by the algorithms to converge to a policy on a fixed number of samples, and (3) converge faster to a near-optimal policy given a large number of samples.

Categories and Subject Descriptors

I.2.6 [**Learning**]: Miscellaneous

General Terms

Algorithms, Performance

Keywords

Transfer Learning, Reinforcement Learning, Sparse Coding, Inter-task mapping, Sparse Gaussian Processes, Optimization

1. INTRODUCTION

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

Transfer learning is the field that studies how to effectively leverage knowledge learned from one or more source tasks when learning one or more target tasks. Although TL has been widely studied within the supervised learning framework (e.g., [9, 10]), TL in RL domains only recently started to gain interest and is very much in flux [12, 18, 21].

Reinforcement learning (RL) is a popular framework that allows agents to solve sequential decision making problems with minimal feedback. Unfortunately, RL agents may learn slowly in large or complex environments due to the computational effort needed to converge to an acceptable performing policy. TL is one technique that copes with this difficulty by providing a good starting policy or prior for the RL agent.

Typically, the source and target task are different but related. In RL settings, the source and target tasks have different representations of state and action spaces, requiring a mapping between the tasks. An inter-task mapping matches each state/action pair of the source task to its correspondance in the target task. Two issues stand out in current TL for RL research. First, although there have been a number of successes in using an inter-task mapping for TL, the mappings are typically *hand-coded* and may require substantial human knowledge [19]. Two fundamental open questions are, first, to what extent it is possible to learn the mapping automatically and, second, how should an inter-task mapping be best leveraged to produce successful transfer?

We tackle these problems and make a number of contributions. The primary contribution is a novel method to automatically learn an inter-task mapping between two tasks based on sparse coding, sparse projection learning, and sparse Gaussian processes. More specifically, we define an inter-task mapping to be a function that relates state-action successor state triplets from the source task to the target task. This inter-task mapping is more than a one-to-one mapping between the state and/or action spaces of the MDPs, as it also includes non-linear terms that are automatically discovered by global approximators, which ultimately enhance the efficacy of transfer. This inter-task learning framework can be split into three essential parts. The first is a dimensional mapping of both the source and target task state-action spaces of the MDPs. The second is the automatic discovery of a high dimensional informative space of the source task. This is achieved through sparse coding, as described in Section 4.1.2, ensuring that transfer is conducted in a high representational space of the source task. In order to use a similarity measure among different patterns, the data should be present in the same space, which is why the target task triplets need to

be projected to the high representational space of the source (done via sparse projection learning, described in Section 4.2). The third and final step is to approximate the inter-task mapping via a non-parametric regression technique, explained in Section 4.3.

Another contribution is to introduce two new algorithms for transfer between tasks of continuous state spaces: Transfer Least Squares Policy Iteration (TrLSPI) and Transfer Fitted-Q-Iteration (TrFQI). Experiments illustrate the feasibility and suitability of the presented approach, and furthermore show that this introduced method can successfully transfer between two different RL benchmarks: transfer is successful between the inverted pendulum and the cart pole, as well as between the mountain car and the cart pole.

2. RELATED WORK

This section presents a selection of related work, focused on transfer learning for reinforcement learning tasks.

TL in RL has been of growing interest to the agents community, due in part to its many empirical successes at significantly improving the speed and/or quality of learning [18]. However, the majority of existing work assumes that 1) the source task and target task are similar enough that no mapping is needed, or 2) an inter-task mapping is provided to the agent.

For example, many authors have considered transfer between two agents which are similar enough that learned knowledge in the source task can be directly used in the target task. For instance, the source and target task could have different reward functions (e.g., *compositional learning* [13]) or have different transition functions (e.g., changing the length of a pole over time in the cart pole task [12]). More difficult are cases in which the source task and target task agents have different state descriptions or actions. Some researchers have attempted to allow transfer between such agents without using an inter-task mapping. For example, a shared *agent space* [3] may allow transfer between such pairs of agents, but requires the agents to share the same set of actions, and requires an agent-centric mapping. Other approaches assume that an inter-task mapping is provided, such as Torrey *et al.* [21] who transfer advice between agents and Taylor *et al.* [19] who transfer Q-value functions by leveraging an existing inter-task mapping.

The primary contrast between these methods and the current work is that we are interested in *learning* a mapping between states and actions in pairs of tasks, rather than assuming that it is provided, or rendered unnecessary because of similarities between source task and target task agents, a requirement for fully autonomous transfer.

There has been some recent work on learning such mappings. For example, semantic knowledge about state features between two tasks may be used [4, 8], background knowledge about the range or type of state variables can be used [16, 20], or transition models for each possible mapping could be generated and tested [17]. However, there are currently no general methods to learn an inter-task mapping without requiring 1) background knowledge that is not typically present in RL settings, or 2) an expensive analysis of an exponential number (in the size of the action and state variable sets) of inter-task mappings. This paper overcomes these problems by automatically discovering high-level features and using them to transfer knowledge between agents without suffering from an exponential explosion. Others have focused on transferring samples between tasks. For instance, Lazaric et al. [6] transfers samples in batch reinforcement learning using a compliance measure. The main difference to this work is that we neither assume any similarities between the transition probabilities, nor restrict the framework to similar state and/or action feature representations.

In contrast to all existing methods (to the best of our knowledge), this paper allows for differences between all variables describing Markov decision processes for the source and target tasks and learns an *inter-task mapping*, rather than a mapping based on state features. Furthermore, the framework introduced in this paper can use *state-dependent* action mappings, allowing additional flexibility.

3. PRELIMINARIES

This section briefly covers reinforcement learning and sparse coding, introducing the necessary notation for the rest of the paper. Section 4 will draw the connection between the two and introduce the main contribution of the paper.

3.1 Reinforcement Learning (RL)

In an RL problem, an agent must decide how to sequentially select actions to maximize its expected return [1, 15]. Such problems are typically formalized as Markov decision processes (MDPs), defined by $\langle S, A, P, R, \gamma \rangle$. S is the (potentially infinite) set of states, A is the set possible actions that the agent may execute, $P : S \times A \times S \to [0, 1]$ is a state transition probability function, describing the task dynamics, $R : S \times A \times S \to \mathbb{R}$ is the reward function measuring the performance of the agent, and $\gamma \in [0, 1)$ is the discount factor. A policy $\pi : S \times A \to [0, 1]$ is defined as a probability distribution over state action pairs, where $\pi(s, a)$ represent the probability of selecting action a in state s. The goal of an RL agent is to improve its policy, potentially reaching the optimal policy $\pi^\star$ which maximizes cumulative future rewards. It can be attained by taking greedy actions according to the optimal Q-function $Q^\star(s, a) = \max_\pi E[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) | s = s_0, a = a_0]$. In tasks with continuous state and/or action spaces, Q and π cannot be represented in a table format, typically requiring sampling and function approximation techniques. This paper uses two such techniques, *Least Squares Policy Iteration* (LSPI) and *Fitted-Q-Iteration* (FQI), discussed later.

3.2 Sparse Coding

Sparse coding (SC) [7] is an unsupervised feature extraction technique that finds a high-level representation for a set of unlabeled input data by discovering a succinct, over-complete basis for the provided data set.

Given a set of m k-dimensional vectors, $\boldsymbol{\zeta}$, SC aims to find a set of n basis vectors, $\mathbf{b}$, and activations, a, with $n > k$ such that $\boldsymbol{\zeta}^{(i)} \approx \sum_{j=1}^{n} a_j^{(i)} \mathbf{b}_j$, where i and j represent the number of input data patterns and number of bases, respectively. SC begins by assuming a Gaussian and a sparse prior on the reconstruction error $(\boldsymbol{\zeta}^{(i)} - \sum_{j=1}^{n} a_j^{(i)} \mathbf{b}_j)$ and on the activations, solving the following optimization problem:

$$\min_{\{\mathbf{b}_j\}, \{a_j^{(i)}\}} \sum_{i=1}^{m} \frac{1}{2\sigma^2} ||\boldsymbol{\zeta}^{(i)} - \sum_{j=1}^{n} \mathbf{b}_j a_j^{(i)}||_2^2 + \beta \sum_{i=1}^{m} \sum_{j=1}^{n} ||a_j^{(i)}||_1 \quad (1)$$

$$s.t. \ ||\mathbf{b}_j||_2^2 \leq c, \forall j = \{1, 2, \ldots, n\}$$

The problem presented in Equation 1 is considered to be a "hard" optimization problem as it is not jointly convex (in the activations and bases). However, fast and efficient optimization algorithms exist [7] and were used, as described in Section 4.1.

4. LEARNING AN INTER-TASK MAPPING

In this section, we cast the problem of learning the inter-task mapping, χ, as a supervised learning problem. We define χ to be a mapping of state-action-state triplets from the source task to the target task. Learning such a mapping requires related triplets from both tasks as data points for training. Unfortunately, obtaining such corresponding triplets is itself a hard problem — it is not trivial for a user to describe which triplets in the source task correspond to which triplets in the target task.

To automatically construct these data points, we propose a novel framework utilizing sparse coding together with an L_1 projection scheme.

We approach the problem by automatically transforming the source task space (i.e., state-action-state space) into a higher representational space through SC, followed by a projection of the target task triplets onto the attained bases. We then use a simple Euclidean distance measure[1] to gauge similarity (Section 4.1). At this stage, the data set is ready to be provided to the learning algorithm so that it may construct the inter-task mapping.

The following sections further clarify each of the above steps and explain the technicalities involved.

4.1 Sparse Coding Transfer for RL

As described in Section 3.2, SC is an efficient way to discover high-level features in an unlabeled data set. First, SC learns how best to match the dimensions of the two different MDPs. Second, SC discovers higher-level features for the low dimensional task's state-action-state space.

4.1.1 Mapping the Source and Target Dimensions

To generate triplet matchings through SC, the first step is to match the dimensions of the state-action-state spaces of the source and target MDPs, which are likely to be different. In principle, after this step any existing TL in RL technique can be used. However, this paper goes further and proposes a new transfer framework based on the discovered bases and activations, described in Section 5.

This "dimensional mapping" process is summarized in Algorithm 1. In short, it sparse codes random triplets $\langle s_0, a_0, s_0^{'} \rangle$ from the task with the lowest dimensionality, constrained to learn a number of bases (d_1) equal to the higher dimension of the unmodified task (regardless of which is the source and target task). We use existing algorithms [7] to solve the Equation from step 2 of Algorithm 1.

4.1.2 High Information Representation

After mapping the source and target dimensions as described in the previous section, SC is again used to discover a succinct higher feature bases of the activations than the unified dimensional spaces that were discovered in Section 4.1.1. If successful this step will discover new features in the source task that could better represent relations with the target task than the bases discovered in Section 4.1.1. Algorithm 2, similar in spirit to Algorithm 1, describes this process.

Algorithm 2 sparse codes the activations, which represent the original source task triplets of the MDPs, to a higher representational space, d_n.[2] This stage should guarantee that we project the triplets of the source task MDP into a high feature space where a similarity measure can be used to find a relation between the source and target task triplets. Note that there are no restrictions on the number of bases: unneeded bases will end up with an activation of zero.

Algorithm 2 discovers new features in the source or target state-action-state spaces. As TL typically transfers from a low dimensional source task to a high dimensional target task, SC determines new bases that are of a higher dimensionality than the original representation used for states and actions in the source task. These newly discovered bases can describe features not anticipated in the original design of the MDP's representation. These new features can highlight similarities between the source and target task thus helping and guiding the transfer learning scheme. The re-encoded triplets—described as a linear combinations of the bases and activations (i.e., **AB**)—do not yet relate to the triplets of the other task. The target task triplets still need to be projected towards these new sparse coded source task bases features. This is done as described in Section 4.2.

Algorithm 1 Sparse Coding TL for RL

Require: Triplets $\{\langle s_0, a_0, s_0^{'} \rangle\}_{i=1}^{m}$ and $\{\langle s_1, a_1, s_1^{'} \rangle\}_{j=1}^{f}$ from both MDPs

1: Determine d_0 and d_1, the dimensions of the state-action-state spaces for both MDPs, where $d_0 \leq d_1$

2: Sparse code the lower dimensional triplets by solving:

$$\min_{\{\mathbf{b}_j\},\{a_j^{(i)}\}} \sum_{i=1}^{m} \frac{1}{2\sigma^2} ||\langle s_0, a_0, s_0^{'} \rangle^{(i)} - \sum_{j=1}^{d_1} \mathbf{b}_j a_j^{(i)}||_2^2 + \beta \sum_{i=1}^{m} \sum_{j=1}^{d_1} ||a_j^{(i)}||_1$$

$$s.t.\ ||\mathbf{b}_j||_2^2 \leq c, \forall j = \{1, 2, \ldots, d_1\}$$

3: Solve the above equation by using an existing algorithm [7]

4: Return the Activation matrix ($\mathbf{A} \in \mathbb{R}^{m \times d_1}$) and the Bases ($\mathbf{B} \in \mathbb{R}^{d_1 \times d_0}$)

Algorithm 2 Succinct High Information Representation of MDPs

Require: Activations **A** from Algorithm 1, Target number of new high dimensional bases d_n

1: Represent the activations in the d_n bases by solving the following problem (again using the algorithm in [7]):

2:

$$\min_{\{\mathbf{z}_j\},\{c_j^{(i)}\}} \sum_{i=1}^{m} \frac{1}{2\sigma^2} ||\langle \mathbf{a}_{1:d_1} \rangle^{(i)} - \sum_{j=1}^{d_n} \mathbf{z}_j c_j^{(i)}||_2^2 + \beta \sum_{i=1}^{m} \sum_{j=1}^{d_n} ||c_j^{(i)}||_1$$

$$s.t.\ ||\mathbf{z}_j||_2^2 \leq o, \forall j = \{1, 2, \ldots, d_n\}$$

3: **return** Return new activations $\mathbf{C} \in \mathbb{R}^{m \times d_n}$ and bases $\mathbf{Z} \in \mathbb{R}^{d_n \times d_1}$

4.2 L_1 Sparse Projection Learning

Once the above stages have finished, the source task triplets are described via the activations **C** (generated in Algorithm 2). However, target task triplets still have no relationship to the learned activations. Since we are seeking a similarity correspondence between

[1] A simple Euclidian distance might not be optimal, but optimizing this measure is planned as future research. Experiments show that more than reasonable results can be attained using this simple approximation.

[2] In our experiments we have set d_n to be 100, a relatively high number.

the source and target task triplets, the target task triplets should be represented in the same high informational space of the source task. Therefore, the next step is to learn how to project the target task triplets onto the **Z** basis representation. The overall scheme is described in Algorithm 3, where the activations are learned by solving the L_1 regularized least squares optimization problem in step 2. This optimization problem guarantees that the activations are as sparse as possible and is solved using the interior point method [2]. The next step will be to order the data points from both the source and the target tasks, which are then used to approximate the inter-task mapping.[3]

4.3 Approximating an Inter-Task Mapping

To finalize the problem of approximating χ, corresponding triplets from the source and target task should be provided to a regressor. We approach this problem by using a similarity measure in the high feature space, **Z**, to identify similar triplets from the two tasks. This similarity measure identifies triplets from the source task that are most similar to those of the target task, and then map them together as being inputs and outputs for the regression algorithm, respectively, as shown on line 2 of Algorithm 4. Since the similarity measure is used in the sparse coded spaces, the distance is calculated using the attained activation (**C** and **Φ**) rather than the triplets themselves. Therefore, the scheme has to trace the data back to the original dimensions of the state-action pairs of the MDPs.

Algorithm 3 Mapping Target Task Triplets

Require: Sparse Coded Bases **Z** generated by Algorithm 2, Target MDP triplets $\{\langle s_t, a_t, s_t^{'}\rangle\}_{i=1}^{f}$

1: **for** $i = 1 \rightarrow f$ **do**

2: Represent the target data patterns in the sparse coded bases, **Z**, by solving:

$$\hat{\phi}^{(i)}(\langle s_t, a_t, s_t^{'}\rangle) = \arg\min_{\phi^{(i)}} ||\langle s_t, a_t, s_t^{'}\rangle^{(i)} - \sum_{j=1}^{d_n} \phi_j^{(i)} \mathbf{z}_j||_2^2 + \beta||\phi^{(i)}||_1$$

3: **return** Activations **Φ**

Algorithm 4 Similarity Measure & Inter-Task mapping approximation

Require: Sparse Coded Basis **Z**, Sparse Coded Activations of the source task $\mathbf{C} \in \mathbb{R}^{m \times d_n}$, Projected Target Task activations $\mathbf{\Phi} \in \mathbb{R}^{m \times d_n}$

1: **for all** ϕ **do**

2: Calculate the closest activation in **C** minimizing the Euclidean/similarity distance measure.

3: Create a data set $\mathcal{D}$ from minimum-distance triplets

4: Approximate the Inter-task mapping, χ, from $\mathcal{D}$ with an appropriate learning algorithm

5: **return** The approximated Inter-task mapping, χ

5. TRANSFER SCHEME

This section describes the novel transfer scheme.

5.1 Implementation Details

Here we introduce implementation details and background used in Section 5.2. Due to space constraints, we briefly describe Least Squares Policy Iteration (LSPI), Fitted-Q-Iteration (FQI) and Gaussian Processes (GPs).

5.1.1 Least Squares Policy Iteration

LSPI [5] is an approximate RL algorithm using the actor/critic framework. LSPI is composed of two parts: the policy is evaluated with *Least Squares Temporal Difference Q-learning* (LSTDQ) and then it is improved. LSTDQ is used to update the weights that parameterize the policy to minimize an error criterion. Once this step has finished, LSPI uses the weights to improve the policy by taking greedy actions with respect to the new Q-function.

5.1.2 Fitted-Q-Iteration

FQI [1] is another approximate RL technique that works by approximating the Q-function as a linear combination of weights and state-action basis functions. FQI operates within two spaces: 1) the parameter space and 2) the Q-function space. During each iteration of the algorithm, the Q-function is updated via the Bellman operator in its corresponding space. Then the function is projected back to the parameter space. These steps are repeated to find high quality policies in practice, although convergence to an optimal Q-function is not guaranteed.

5.1.3 Gaussian Processes

GPs are supervised learning techniques used to discover a relation between a given set of input vectors, $\mathbf{x}$, and output pairs, $\mathbf{y}$. A full mathematical treatment can be found elsewhere [11, 14]. Unlike many regression techniques, which perform inference in the weight space, GPs perform inference directly in the function space. Learning in a GP setting involves maximizing the marginal likelihood to find the hyper-parameters best describing the data. Maximizing the likelihood may be computationally complex. Therefore, we use a fast learning technique, *Sparse Pseudo-input Gaussian Processes* (SPGP) [14], to quickly model the complex inter-task mapping.

5.1.4 Sparse Pseudo-Inputs Gaussian Processes

SPGPs aim to reduce the complexity of learning and prediction in GPs by parametrizing the regression model with $M \ll N$ (N is the number of input points) pseudo-input points, while still preserving the full Bayesian framework. The covariance of the GP model is parametrized by the location of the $M \ll N$ pseudo-inputs and training aims at finding the parameters and the locations of the pseudo-points that best describe the data. Existing results [14] show a complexity reduction in the training cost (i.e., the cost of finding the parameters of the covariances) and in the prediction cost (i.e., prediction on a new set of inputs) compared to GP regression. The results further demonstrate that the SPGP framework can match normal GPs approximation power with small M (i.e., few pseudo-inputs).

5.2 Transfer Details

This section proposes two novel transfer algorithms for pairs of tasks with continuous state spaces and discrete action spaces, titled Transfer Least Squares Policy Iteration (TrLSPI) and Transfer Fitted-Q-Iteration (TrFQI), which makes use of a learned source task policy.

Transfer Least Squares Policy Iteration.

TrLSPI is described in Algorithm 5 and can be applied to any TL in RL problem having continuous states and discrete action spaces. It is also sample efficient as it preserves the advantages of the normal LSPI algorithm. TrLSPI can be split into two sections. The first (lines 1–4 of Algorithm 5) determine χ (see Section 4), using

[3]We use the s and t indices to describe the source task (typically of lower dimensions) and the target task.

Algorithm 5 TrLSPI

Require: Source MDP triplets $\{\langle s_s, a_s, s_s^{'}\rangle\}_{i=1}^{m}$, Target MDP triplets $\{\langle s_t, a_t, s_t^{'}\rangle\}_{j=1}^{f}$, Number for re-samples n_s, close to optimal policy for the source system $\pi_s^\star$, State action basis functions for the target task $\psi_1, \ldots, \psi_k$

1: *Map the Dimensions* using Algorithm 1
2: Discover *High Informational Representation* using Algorithm 2
3: *Sparse Project* the target task triplets using Algorithm 3
4: Use a *similarity* measure to attain the data set and *approximate* χ using Algorithm 4
5: Randomly sample n_s source task triplets $\langle s_s, a_s, s_s^{'}\rangle_{i=1}^{n_s}$ greedily in the optimal policy $\pi_s^\star$, set of state-dependent basis function $\psi_1, \ldots, \psi_k : S_t \times A_t \to \mathbb{R}$
6: **for** $i = 1 \to n_s$ **do**
7: Find the corresponding target task triplets as $\langle s_t^{(i)}, a_t^{(i)}, s_t^{(i)\prime}\rangle = \chi(\langle s_s^{(i)}, a_s^{(i)}, s_s^{(i)\prime}\rangle)$
8: Use the black box generative model of the environment to produce the rewards on the transferred triplets
9: Use LSTDQ to evaluate transformed triplets
10: Improve policy until convergence using LSPI
11: **return** Learned policy $\pi_t^\star$

source and target task triplets.[4] The second section (lines 5–9) provides triplets (using $\pi_s^\star$) as a start for the evaluation phase of the LSPI algorithm (LSTDQ), allowing it to improve the target task policy. If the tasks are similar, and if the inter-task mapping is "good enough," then those triplets will 1) bias the target task controller towards choosing good actions and 2) restrict its area of exploration, both of which help to reduce learning times and increase performance.

Algorithm 5 leverages source task triplets to attain a good starting behavior for the target task. The performance of the policy necessarily depends on the state space region where those triplets were provided. In other words, it is not possible to achieve near-optimal performance with a small number of triplets that are in regions far from the goal state.[5] Therefore, to learn a near-optimal policy, it must collect triplets from the target with the current policy, or a large number of source task triplets should be provided. A full model of the system is not required but the algorithm does require a black box generative model for sampling.

Transfer Fitted-Q-Iteration.

The second novel algorithm, Transfer Fitted-Q Iteration (TrFQI), is also capable of transferring between MDPs with continuous state space and countable actions and preserves the advantages of the standard FQI algorithm. The key idea is to provide a good starting sample distribution on which the FQI algorithm can learn. This distribution is provided to the target task agent via χ, which in turn maps the source task triplets (sampled according to $\pi_s^\star$) into the target task. Algorithm 6 presents the pseudocode and can also be split into two parts. First, lines 1-8 use the inter-task mapping to project the source task triplets to the target task (same steps followed in the first part of TrLSPI). Second, lines 9-10 provide those triplets to the FQI Algorithm to learn a policy.[6]

[4]The source task policy may be optimal or near-optimal, depending on the RL algorithm used in the source task.

[5]This is a problem that is inherit to LSPI and not a property of the TrLSPI algorithm.

[6]It is also worth noting that the generalization of this algorithm depends on the type of function approximators used to approximate the Q-function. This is a property of FQI and not of TrFQI. Therefore, if the algorithm has to attain near-optimal behavior, either a large amount of triplets should be provided, or it must again have access to a black box generative model of the MDP for re-sampling.

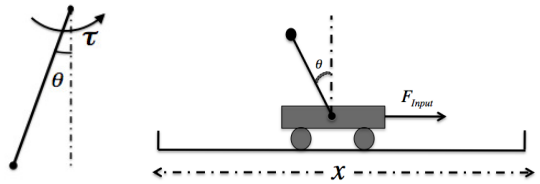
Algorithm 6 TrFQI

Require: Source MDP triplets $\{\langle s_s, a_s, s_s^{'}\rangle\}_{i=1}^{m}$, Target MDP triplets $\{\langle s_t, a_t, s_t^{'}\rangle\}_{j=1}^{f}$, Number for re-samples n_s, close to optimal policy for the source system $\pi_s^\star$, State action basis functions for the target task $\psi_1, \ldots, \psi_k$

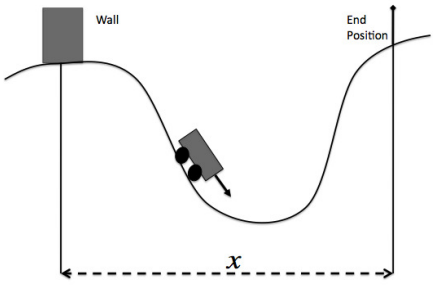
1: *Map the Dimensions* using Algorithm 1
2: Discover *High Informational Representation* using Algorithm 2
3: *Sparse Project* the target task triplets using Algorithm 3
4: Use a *similarity* measure to attain the data set and *approximate* χ using Algorithm 4
5: Randomly Sample n_s Source task triplets $\langle s_s, a_s, s_s^{'}\rangle_{i=1}^{n_s}$ greedily in the optimal policy $\pi_s^\star$, set of state-dependent basis function $\psi_1, \ldots, \psi_k : S_t \times A_t \to \mathbb{R}$
6: **for** $i = 1 \to n_s$ **do**
7: Find the corresponding target task triplets as $\langle s_t^{(i)}, a_t^{(i)}, s_t^{(i)\prime}\rangle = \chi(\langle s_s^{(i)}, a_s^{(i)}, s_s^{(i)\prime}\rangle)$
8: Use the black box generative model of the environment to produce the rewards on the transferred triplets
9: Apply FQI
10: **return** Learned policy $\pi_t^\star$

6. EXPERIMENTS & RESULTS

We have conducted two experiments to evaluate the framework. The first was the transfer from the Inverted Pendulum (IP), Figure 1(a), to the Cart Pole (CP), Figure 1(b). The second experiment transfers between Mountain Car (MC), Figure 1(c) and CP. This section describes the experiments conducted.



(a) Inverted Pendulum (b) Cart Pole



(c) Mountain Car

Figure 1: Experimental domains

6.1 Inverted Pendulum to Cart Pole Transfer

The source task was the inverted pendulum problem. The state variables describing the systems are the angle and angular velocity $\{\theta,\ \dot{\theta}\}$. The control objective of the IP is to balance the pendulum in an upright position with an angle, $\theta = 0$ and angular velocity $\dot{\theta} = 0$. The allowed torques are $+50$, 0 and $-50\ Nm$. The reward function is $cos(\theta)$ which yields its maximum value of $+1$ at the up-right position.

In cart pole, the goal is to swing up the pole and keep it balanced in the upright position (i.e., $\theta = \dot{\theta} = 0$). The allowed actions are (+1) for full throttle right and (-1) for full throttle left. The reward function of the system consisted of two parts: (1) $cos(\theta)$, which yields its maximum value of $+1$ at the upright position of the pole, and (2) -1 if the cart hits the boundaries of the track. The angle and position were restricted to be within $|\theta| < \frac{\pi}{9}$ and $|x| < 3$.

In order to transfer between IP and CP, we first learn an optimal policy in the source task, $\pi^{\star}_{IP}$, with LSPI. $\pi^{\star}_{IP}$ was then used to randomly sample different numbers of initial states of task, to be used by χ. We started with 5000 and 2500 randomly sampled states (using a random policy) for the IP and the CP, respectively. These triplets were used by the algorithm described in Section 4 to learn the inter-task mapping χ[7]. After χ had been learned, different numbers of samples were collected from the source task using $\pi^{\star}_{IP}$. Specifically, we have sampled $500, 1000, \ldots 20000$ states as input to the TrLSPI and the TrFQI algorithms to measure performance and convergence times.

6.1.1 TrLSPI Results

Our results show both an increase in the performance on a fixed number of samples and a decrease in the convergence times in both a predefined number of samples and to attain an optimal policy. We measured the performance as the number of steps during an episode to control the pole in an upright position on a given fixed amount of samples. Figure 2 summarizes the results attained on a different number of transferred samples and compares them with those attained through normal LSPI learning scheme. It shows an increase in the number of control steps (i.e., steps the pole was in an up-right position) in the case of the transferred samples compared to a random sampling scheme. It can be seen that when using 2000 samples (i.e., a small number of samples) our transfer scheme was able to attain an average of 520 control steps while random initialization reached only 400. This performance increases with the number of samples to reach 1200 steps at 10000 transferred samples.

Another measure was the time required to learn a near-optimal target task policy using only a fixed number of samples. There was a decrease in the convergence times, represented by the number of iterations in LSPI, when provided a fixed amount of transferred samples. LSPI was able to converge faster once provided the transferred samples compared to a random sample data set. For example, it took LSPI 5 iteration to converge provided 5000 transferred samples but 7 iterations in the random case. Further, the algorithm converged within 12 iteration provided 20000 transferred samples while it took it about 19 for the random case. Finally, LSPI was able to converge to an acceptable policy (i.e., an 800 control steps) within 22.5 minutes after being provided a random data set, compared to 16 minutes with the transferred data set[8]. Calculating χ took an additional 3.7 minutes.

[7]We believe but have not confirmed that the samples to learn χ should be provided using random policies in both the source and the target task as we need to cover most large areas of the state-action spaces in both tasks.

[8]Our experiments were performed on a 2.8 Ghz Intel Core i7.

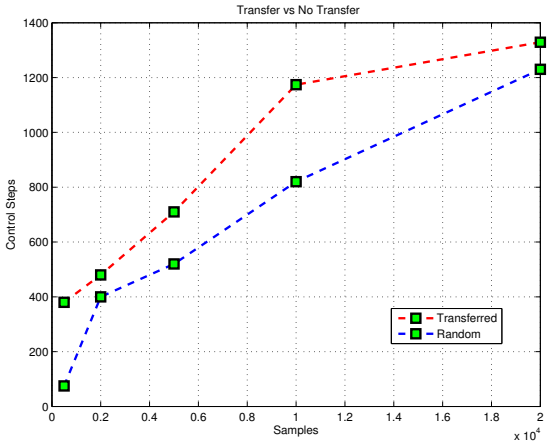


Figure 2: Cart Pole results from LSPI and TrLSPI after learning on Inverted Pendulum: the performance is measured after collecting 500 different initial states in the target tasks.

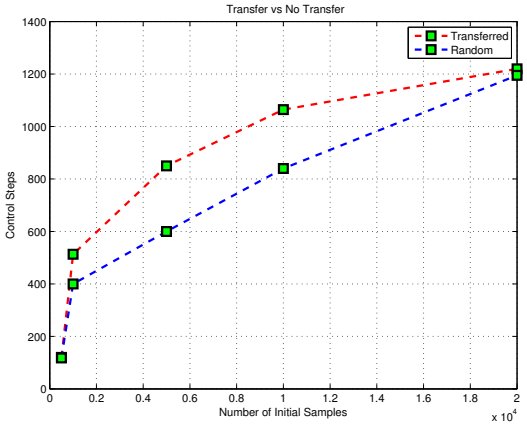


Figure 3: Cart Pole results with FQI and TrFQI after learning on Inverted Pendulum: the performance is measured after collecting 500 different initial states in the target tasks.

6.1.2 TrFQI Results

We performed similar experiments using the other proposed algorithm TrFQI. Similar results were observed as could be seen from Figure 5. The transferred samples produce more control steps on the target task compared to learning on random samples. As an illustration, the algorithm was able to achieve 800 control steps when using 5000 transferred states but it needed about 10000 random samples to attain the same performance. We also report a decrease in the number of training iterations in the TrFQI compared to FQI at a fixed number of samples and to attain an optimal policy. We have observed good performance at 50 iteration of training on transferred samples compared to 70 iterations for the random case. Moreover, TrFQI was able to reach a suboptimal acceptable policy with about 85 iteration once using transferred samples compared to a 150 iterations for the random case.

6.2 Mountain Car to Cart Pole Transfer

The source task was the MC problem, a benchmark RL task. The car has to drive up the hill (Figure 1(c)). The difficulty is that gravity is stronger than the car's motor—even at maximum throttle

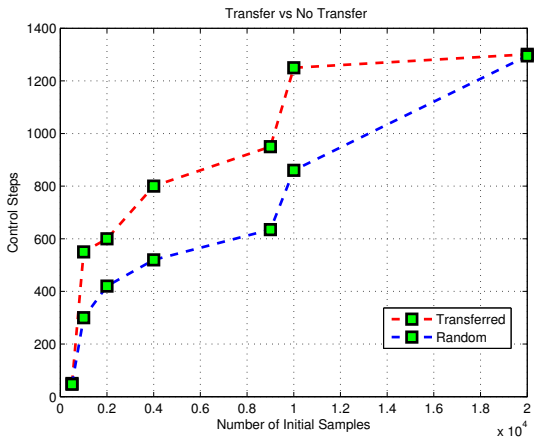


Figure 4: Cart Pole results using LSIP and TrLSPI after learning on Mountain Car: the performance is measured after collecting 500 different initial states in the target tasks.

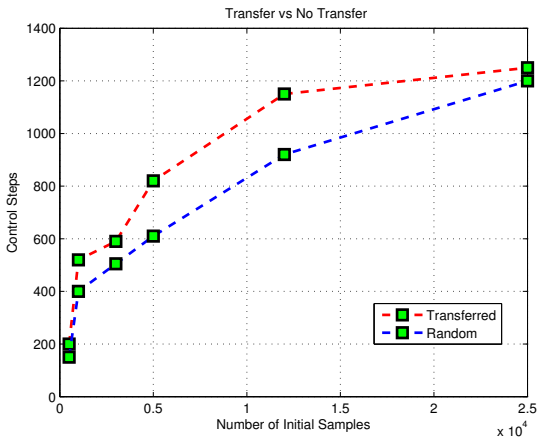


Figure 5: Transfer Results on the Cart Pole task using TrFQI after learning on Mountain Car: the performance is measured after collecting 500 different initial states in the target tasks.

the car can not directly reach the top of the hill. The dynamics of the car are described via two continuous state variables (x, $\dot{x}$) representing the position and velocity of the center of gravity of the car, respectively. The input action takes on three distinct values: maximum throttle forward (+1), zero throttle (0), and maximum throttle reverse (-1). The car is rewarded by $+1$ once it reaches the top of the hill, -1 if it hits the wall, and zero elsewhere.

The target task is the Cart Pole problem, as described in the previous experiment.

SARSA(λ) [15] is used to learn $\pi^{\star}_{MC}$ in the source task. The policy was then used to randomly sample different numbers of source task states, to be used by χ. We started with 5000 and 2500 randomly sampled states for the Mountain Car and the Cart Pole, respectively. These samples were used by the algorithm described in Section 4 to learn the inter-task mapping χ. After χ has been learned, different numbers of samples were collected from the source task using $\pi^{\star}_{MC}$. Specifically, we have sampled $500, 1000, \ldots 20000$ states as input to the TrLSPI and the TrFQI algorithms to measure performance and convergence times.

6.2.1 *TrLSPI Results*

Figure 4 clearly shows an increase in the number of control steps in the case of the transferred samples compared to a random sampling scheme. When using 2000 samples, our transfer scheme was able to attain an average of 600 control steps. Achieving a similar performance required roughly 4000 random samples. This performance increases with the number of samples to finally reach about 1300 control step on 20000 samples for both cases. We also report a decrease in the convergence times, represented by the number of iterations in LSPI, provided a fixed amount of transferred samples. LSPI was able to converge faster once provided the transferred samples compared to a random sample data set. For example, it took LSPI 7 iteration to converge provided 5000 transferred samples but 12 iterations in the random case. Further the algorithm converged within 14 iterations provided 20000 transferred samples while it took it about 19 for the random case. Finally, LSPI was able to converge to an acceptable policy within a 22.5 minutes after being provided a random data set, compared to 17 minutes with the transferred data set. Calculating χ took an addition 3.7 minutes.

6.2.2 *TrFQI Results*

The analogous experiments using TrFQI produced similar results, as shown in Figure 5. Transferred samples where able to produce a higher number of control steps in the target task when compared to learning on random samples in the target tasks. As an illustration, the algorithm was able to attain a performance of 800 control steps when using 5000 transferred states, but needed 9000 random samples to attain the same threshold. Using transferred and random samples both allow FQI to converge to roughly the same performance (1200) when provided a large number of samples. We also report a decrease in the number of training iterations at a fixed number of samples and to attain an optimal policy. We have observed good performance at 50 iteration of training on transferred samples compared to 80 iterations for random samples. Moreover, TrFQI was able to reach a suboptimal policy with about 91 iteration once using transferred samples compared to a 150 iterations for the random case.

7. ANALYSIS & DISCUSSION

It is clear from the results presented that the learner's performance increased using our proposed framework, relative to a random selection scheme. Policy performance improved, as measured by the number of control steps achieved by the agent on the target task. The number of learning iterations required also decreased, as measured by the number of iterations required by the algorithm to converge to a policy on a fixed number of transferred samples. This leads us to conclude that TrFQI and TrLSPI both:

1. provided a better distribution of samples compared to random policy in the target task,

2. required fewer iterations to converge to a fixed policy when provided a fixed number of transferred samples, and

3. reached a near-optimal performance policy faster than when using random selection scheme.

Furthermore, these results show that the proposed framework

4. successfully learned an inter-task mapping between two sets of different RL tasks.

We speculate that the framework is applicable to any model-free TL in RL problem with continuous state spaces and discrete action spaces, covering many real world RL problems. The framework has the advantage of automatically finding the inter-task functional mapping using SC and any "good" regression technique. One potential weakness is that our framework should work correctly when the two tasks at hand are *semantically* similar, as the rewards of the two systems were not taken into account in the explained scheme. For instance, consider the transfer example between the cart pole and "cart fall" tasks. The control goals of these two tasks are opposite whereby in the cart pole the pole has to be balanced in the upright position while in the cart fall the pole has to be dropped as fast as possible. In other words, the agents have the same transitions in the two tasks but have to reach two opposite goal. Our mapping scheme of Section 4, once applied, will produce a one-to-one mapping from the source to the target task relating the same transitions from both of the tasks together. Clearly the optimal policies of the two tasks are opposite. In this case the target task would be provided with a poor bias, potentially hurting the learner (i.e., producing negative transfer). We think that our approach will be able to avoid this scheme once the rewards are added to the similarity measure generating the training set to approximate the inter-task mapping χ, but such investigation is left to future work.

8. CONCLUSIONS & FUTURE WORK

This paper has presented a novel technique for transfer learning in reinforcement learning tasks. Our framework may be applied to pairs of reinforcement learning problems with continuous state spaces and discrete action spaces. The main contributions of this paper are (1) the novel method of automatically attaining the inter-task mapping, χ and (2) the new TrLSPI and TrFQI algorithms for tasks with continuous state spaces and discrete actions. We approached the problem by framing the approximation of the inter-task mapping as a supervised learning problem that was solved using sparse pseudo input Gaussian processes. Sparse coding, accompanied with a similarity measure, was used to determine the data set required by the regressor for approximating χ. Our results demonstrate successful transfer between two similar tasks, inverted pendulum to cart pole, and two very different tasks, mountain car to cart pole task. Success was measured both in an increase in learning performance as well as a reduction in convergence time. We speculate that the process usefully restricts exploration in the target task and that the transferred state quality resulting from our scheme.

There are many exciting directions for future work. First, we intend to compare different distance metrics and demonstrate their effects on the overall performance of the algorithm. Second, the distance measure will be improved by incorporating the rewards in the framework, helping to avoid the problem of negative transfer.

9. REFERENCES

[1] L. Buşoniu, R. Babuška, B. De Schutter, and D. Ernst. *Reinforcement Learning and Dynamic Programming Using Function Approximators*. CRC Press, Boca Raton, Florida, 2010.

[2] S. jean Kim, K. Koh, M. Lustig, S. Boyd, and D. Gorinevsky. An interior-point method for large-scale l1-regularized logistic regression. *Journal of Machine Learning Research*, 2007, 2007.

[3] G. Konidaris and A. Barto. Autonomous shaping: knowledge transfer in reinforcement learning. In *In Proceedings of the 23rd Internation Conference on Machine Learning*, pages 489–496, 2006.

[4] G. Kuhlmann and P. Stone. Graph-based domain mapping for transfer learning in general games. In *Proceedings of The Eighteenth European Conference on Machine Learning*, September 2007.

[5] M. G. Lagoudakis and R. Parr. Least-squares policy iteration. *J. Mach. Learn. Res.*, 4:1107–1149, December 2003.

[6] A. Lazaric, M. Restelli, and A. Bonarini. Transfer of samples in batch reinforcement learning. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pages 544–551, New York, NY, USA, 2008. ACM.

[7] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *In NIPS*, pages 801–808. NIPS, 2007.

[8] Y. Liu and P. Stone. Value-function-based transfer for reinforcement learning using structure mapping. In *Proceedings of the Twenty-First National Conference on Artificial Intelligence*, pages 415–20, July 2006.

[9] S. J. Pan and Q. Yang. A survey on transfer learning.

[10] R. Raina, A. Battle, H. Lee, B. Packer, and A. Y. Ng. Self-taught learning: Transfer learning from unlabeled data. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, 2007.

[11] C. E. Rasmussen. In *Gaussian processes for machine learning*. MIT Press, 2006.

[12] O. G. Selfridge, R. S. Sutton, and A. G. Barto. Training and tracking in robotics. In *IJCAI*, pages 670–672, 1985.

[13] S. Singh. Transfer of learning by composing solutions of elemental sequential tasks. In *Machine Learning*, pages 323–339, 1992.

[14] E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances In Neural Information Processing Systems*, pages 1257–1264. MIT press, 2006.

[15] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction, 1998.

[16] E. Talvitie and S. Singh. An experts algorithm for transfer learning. In *Proceedings of the Twentieth International Joint Conference on Artificial Intelligence*, 2007.

[17] M. E. Taylor, G. Kuhlmann, and P. Stone. Autonomous transfer for reinforcement learning. In *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 283–290, May 2008.

[18] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *J. Mach. Learn. Res.*, 10:1633–1685, December 2009.

[19] M. E. Taylor, P. Stone, and Y. Liu. Transfer learning via inter-task mappings for temporal difference learning. *Journal of Machine Learning Research*, 8(1):2125–2167, 2007.

[20] M. E. Taylor, S. Whiteson, and P. Stone. Transfer via inter-task mappings in policy search reinforcement learning. In *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 156–163, May 2007.

[21] L. Torrey, T. Walker, J. Shavlik, and R. Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *In Proceedings of the Sixteenth European Conference on Machine Learning*, pages 412–424, 2005.

Learning in a Small World

Arun Tejasvi Chaganty
Deptt. of Computer Science and Engineering,
IIT Madras
Chennai, India - 600036
arunc@cse.iitm.ac.in

Prateek Gaur
Deptt. of Computer Science and Engineering,
IIT Madras
Chennai, India - 600036
prtkgaur@cse.iitm.ac.in

Balaraman Ravindran
Deptt. of Computer Science and Engineering,
IIT Madras
Chennai, India - 600036
ravi@cse.iitm.ac.in

ABSTRACT

Understanding how we are able to perform a diverse set of complex tasks is a central question for the Artificial Intelligence community. A popular approach is to use temporal abstraction as a framework to capture the notion of subtasks. However, this transfers the problem to finding the right subtasks, which is still an open problem. Existing approaches for subtask generation require too much knowledge of the environment, and the abstractions they create can overwhelm the agent. We propose a simple algorithm inspired by small world networks to learn subtasks while solving a task that requires virtually no information of the environment. Additionally, we show that the subtasks we learn can be easily composed by the agent to solve any other task; more formally, we prove that any task can be solved using only a logarithmic combination of these subtasks and primitive actions. Experimental results show that the subtasks we generate outperform other popular subtask generation schemes on standard domains.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning; I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods and Search

General Terms

Algorithms, Theory, Experimentation

Keywords

reinforcement learning, options framework, social network analysis, small world phenomenon

1. INTRODUCTION

Reinforcement learning (RL) is a widely studied learning framework for autonomous agents, particularly because of it's extreme generality; it addresses the problem of learning optimal agent behaviour in an unknown stochastic environment. In this setting, an agent explores a state space, receiving rewards for actions it takes; the objective of the agent is to maximise it's rewards accumulated over time. However, when scaling up to larger domains, these agents require prohibitively large amounts of experience in order to learn a good policy. By allowing the agent to exploit the structure of environment or task, we can reduce the experience required.

Structure can be imposed on a learning task through either spatial or temporal abstractions. With the former, the state-space is minimised using information about the symmetries present in the domain. Spatial abstractions have been surveyed in [6]. In the latter case, high-level actions are introduced which capture sequences of primitive actions. In this light, temporal abstractions capture the notion of a "subtask". The most common approach for temporal abstractions is the options framework proposed by Sutton, Precup and Singh [12], and we build our work on this framework also. Work by Ravindran and Barto on relativised options [11] show how temporal abstractions can be combined with spatial abstractions. Both spatial and temporal abstractions play an important role in transfer learning, where we wish to extend optimal behaviour learnt in one task to another task; a survey of such techniques can be found in [14].

While options provide a broad framework for temporal abstraction, there is still no consensus on how to choose subtasks. The prevalent view is that subtasks should represent skills, i.e. partially defined action policies that constitute a part of many reinforcement learning problems [15]. For this reason, much of the existing work centres around identifying 'bottlenecks', regions that the agent tends to visit frequently [9], either empirically as in [9], or, more recently, using graph theoretic methods like betweenness centrality [2] or graph partitions [10]. The intuition is that options that navigate an agent to such states helps the agent move between strongly connected components, thus leading to efficient exploration.

These option generation schemes suffer from two serious drawbacks; (i) they either require complete knowledge of the MDP or follow a sample-heavy approach of constructing a local model from trajectories, and (ii) there are, in general, several options to bottlenecks that can be initiated by the agent. This leads leading to a blowup in the decision space, often causing the agent to take more time to learn the task as it filters through the unnecessary options.

If one considered these options as additional edges to the bottleneck states, in the sense that a single decision is sufficient to transit the agent from a state, to the bottleneck, the resultant state-interaction graph would now be "more" connected. To highlight the importance of the connectivity of the state-interaction graph, consider the Markov chain induced by a policy for an Markov decision process. It is well

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

known that the convergence rate of a Markov chain (mixing time), is directly related to its conductance [4], and thus its algebraic connectivity.

Recognising the importance of connectivity, we apply concepts from Kleinberg's work on small world networks, to the context of problem solving with autonomous agents. These graphs have been shown to have exceptionally high algebraic connectivity, and thus fast Markov chain mixing times [13]. In a small-world network, each node has one non-neighbouring edge, which connected to another node with a probability inversely proportional to the distance between them. With this simple construction, Kleinberg showed that an agent can discover a short path to any destination using only local information like the coordinates of it's immediate neighbours [5]. In contrast, other graph models with a small diameter only state the existence of a short path, but do not guarantee that an agent would be able to find such a path.

In our context, we construct subtasks distributed according to the small world distribution as follows; create an option that will take the agent from a state s to another state s' with a probability inversely proportional to the distance between s and s'. We prove that this set of subtasks enables the agent to easily solve any task by using only a logarithmic number of options to reach a state of maximal value (Section 3). As this scheme adds at most one additional option per state, we do not explode the decision space for the agent.

Furthermore, in Section 4, we devise an algorithm that learns small world options from the optimal policies learnt over a few tasks in the domain. Thus not only are small world options effective to use, they are also simple to learn, and do not require any global analysis of the MDP. Experiments on several standard domains show that small-world options outperform bottleneck-based methods, and that small world options require significantly fewer learning epochs to be effective.

The remainder of the paper is organised as follows. We present an overview of reinforcement learning, and the options framework in Section 2. We then define a small world option, and prove that given such options, an agent will require to use only a logarithmic number of them to perform a task in Section 3. From a more practical perspective, we present an algorithm to extract these options from optimal policies learnt on several tasks in the domain in Section 4. We present our experimental results in Section 5. Finally, we conclude in Section 6, where we present future directions for our work. Appendix A contains an extension of Kleinberg's proof for the distributed search property of small-world networks which is used in Section 3.

2. BACKGROUND

In reinforcement learning, the standard representation of an environment and task instance is a Markov decision process (MDP). An MDP can be represented as the tuple, $\langle S, A, P, R, \gamma \rangle$, where S and A are finite sets of states and actions, $P : S \times A \times S \to [0, 1]$ describes the dynamics of the world through state-action transition probabilites, $R : S \times A \to \mathbb{R}$ describes the task at hand by ascribing rewards for state transitions, and $\gamma \in [0, 1]$ is a discount factor that weighs the value of future rewards.

In this setting, an agent in a state $s \in S$ chooses an action $a \in A$, and moves to a state s' with probability $P(s, a, s')$, receiving a reward $R(s, s')$. The objective of the agent is to find a policy $\pi : S \times A \to [0, 1]$, i.e. a decision procedure for selecting actions, that maximises the reward it accumulates in the long run, $R = \sum_i \gamma^i r_i$. R is also called the return.

We define the value function $V : S \to \mathbb{R}$ to be the expected return from s, and $Q : S \times A \to \mathbb{R}$ to be the expected return from s, after taking the action a. The optimal value function must satisfy the Bellman optimality equation,

$$\begin{aligned} V(s) &= \max_a R(s,a) + \gamma \sum_{s' \in S} P(s,a,s') V(s') \\ Q(s,a) &= R(s,a) + \gamma \sum_{s' \in S} P(s,a,s') \max_{a'} Q(s',a'). \end{aligned}$$

Given an optimal Q, an agent can construct an optimal policy, $\pi(s, a^*) = 1$ when $a^* = \text{argmax}_a Q(s, a)$, and 0 otherwise. In principle, if the agent knew the MDP, it could construct the optimal value function, and from it an optimal policy. However, in the usual setting, the agent is only aware of the state-action space, S and A, and must learn Q through exploration. The Q-learning algorithm learns Q with a simple update for every step the agent takes,

$$Q(s,a) = Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right],$$

where $\alpha \in [0, 1]$ is a parameter that controls the learning rate. It has been shown that the Q-learning algorithm converges to the optimal value function in the limit with fairly permissive assumptions.

The options framework provides a temporal abstraction through subtasks. An option $\langle \mathcal{I}, \pi, \beta \rangle$ is described by an initiation set $\mathcal{I} \subset S$, a policy π, and a terminating condition β. An agent can exercise an option in any state $s \in \mathcal{I}$, following which, it will follow the policy π described by the option, until the terminating condition $\beta(s)$ is satisfied. The terminating condition β can be stochastic.

Several learning algorithms have been proposed for agents using options [12, 1]. One simple such method that we will use is MacroQ, a generalisation of the Q-learning algorithm described above. The MacroQ algorithm updates the value function only after completion of the option. If the option o was initiated in the state s, and continues for k steps before terminating in s', the corresponding Q function update will be,

$$Q(s,o) = Q(s,o) + \alpha \left[r + \gamma^k \max_{o' \in A \cup \mathcal{O}} Q(s',o') - Q(s,o) \right].$$

Different tasks in the same domain can be described by different R. Let R be sampled from the family $\mathcal{R}$. Our objective then is to find a set of options O that reduces the expected learning time over $\mathcal{R}$.

Example 1. To make the discussion more tangible, let us look at an example, the Taxi domain, shown in Figure 1. The agent is a taxi navigating in this road-map. It must pick up a passenger at one of the 4 pads, R, G, B or Y. Subsequently, it must carry the passenger to a destination, which is also one of the above four pads. The states of the taxi would then be a tuple containing the location of the passenger (in one of the four pads, or within the taxi), the destination of the passenger, and location of the taxi in the map. The actions the taxi can perform are moving up, down, left or right in the map, as well as pick up or drop a passenger at a pad. Typical options for such a domain would be an option that can be started anywhere, and has a policy that takes the taxi to the one of the pads in the

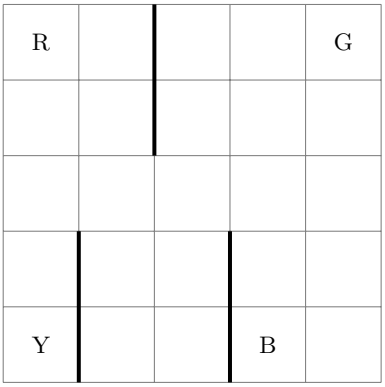


Figure 1: The Taxi Domain

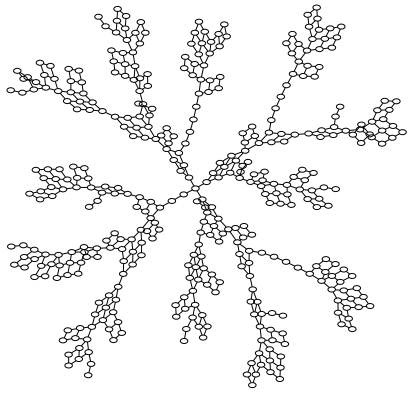


Figure 2: The State Space Graph for Taxi

shortest possible manner. Such an option is generic, and does not depend on where the passenger or destination are. The RL agent must then learn to choose the right option when picking up the passenger.

3. SMALL WORLD OPTIONS

In Kleinberg's small-world network model, each node u is given one 'long-range' edge to a node v, which was chosen with a probability $P_r(u,v) \propto \|u-v\|^{-r}$, where $\|u-v\|$ denotes the least distance between nodes u and v in the graph. Similarly for each state s, we add a single 'path option' to another state s', where s' is chosen with probability $P_r(s,s') \propto \|s-s'\|^{-r}$. A path option $o_p(s,s')$ is an option with $\mathcal{I} = \{s\}$, $\beta = \{s'\}$, and an optimal policy to reach s' for π. Intuitively, it is an option that takes the agent from s to s'. In practice, we may generate path options for only a subset of $|S|$. Note that while this results in $O(|S|)$ options, only one additional option is available in any state, and thus the decision-space for the agent is not significantly larger.

On an r-dimensional lattice, $\mathcal{K}_r$, the distance from any node u to a target node t is bounded by $\|u-t\|$, a quantity which is locally computable. When given long-range edges distributed according to P_r, Kleinberg showed that a greedy distributed algorithm $\mathcal{GA}$ that chooses a neighbour v closest to t will reach t with an expected time $O(\log(|V|)^2)$. This follows as a trivial corollary of the following theorem,

THEOREM 1. *Let $f : V \to \mathbb{R}$ be a function embedded on the graph $\mathcal{G}(V,E)$, such that, $\kappa_1\|u-v\| - c_1 \le \|f(u) - f(v)\| \le \kappa_2\|u-v\| - c_2$, where $0 \le \kappa_1 \le \kappa_2$, and $0 \le c_2 \le \frac{c_1}{2}$. Let M_f be the global maxima of f. Let $\mathcal{GA}_\epsilon$ be an ϵ-greedy algorithm with respect to f, i.e. an algorithm which chooses with probability $1-\epsilon$ to transit to the neighbouring state closest to M_f, i.e. $N(u) = \text{argmin}_v \|f(v) - f(M_f)\|$.*

If $\mathcal{G}(V,E)$ is r-dimensional lattice, and contains a long distance edge distributed P_r, then $\mathcal{GA}_\epsilon$ takes $O((\log|V|)^2)$ steps to reach M_f.

PROOF. The key insight of the proof is that with edges distributed according to P_r, there will be, with high probability, a edge within the neighbourhood of a node to an exponentially smaller neighbourhood of the target. Thus, the agent will only require to hop through $\log|V|$ 'neighbourhoods'. By bounding the time spent in each neighbourhood to $\log|V|$, we arrive at the result. We refer the reader to Appendix A for the complete proof. □

It is easy to construct a graph $\mathcal{G}_M$ out of the state-space described by an MDP. The states S become the nodes of the graph, and actions A become the edges, with the transition probabilities as weights. Options can be viewed as paths along the graph. As an example, the Taxi domain defined earlier translates to the graph shown in Figure 2.

Consider an MDP $M_{\mathcal{K}_r}$ with states connected in a r-dimensional lattice, and noisy navigational actions between states. We claim that by using *robust* path options distributed according to P_r, an ϵ-greedy agent can reach a state of maximal value using $O(\log(|S|)^2)$ options, using the value function V as a local property of the state.

Definition 1. A *robust path option* $o(u,v)$, where $u,v \in S$ is an option that takes the agent from u to v 'robustly', in the sense that in each epoch, the agent moves closer to v with a probability $1-\epsilon > \frac{1}{2}$. [1]. Note that this ϵ includes any environmental effects as well.

The following lemma relates V to the graph distance from the target state, thus allowing us to apply Theorem 1.

LEMMA 1. *Let $o(u,v)$ be the preferred option in state u, and let $\|u-v\|_V = |\log \mathrm{V}(v) - \log \mathrm{V}(u)|$. Then,*

$$k_1\|u-v\| - c_1 \le \|u-v\|_V \le k_2\|u-v\|,$$

where $k_1 = \log\frac{1}{\gamma}$, $k_2 = \log\frac{1}{(1-\epsilon)\gamma}$, and $c_1 = \log\frac{1}{1-\gamma}$.

PROOF. From the Bellman optimality condition, we get the value of $o(u,v)$ to be,

$$\mathrm{Q}\left(u, o\left(u,v\right)\right) \;=\; \mathrm{E}_l\left[\gamma^l \mathrm{V}\left(v\right) + \sum_{i=1}^{l} \gamma^{i-1} r_i\right],$$

where l is the length of the option, and r_i is the reward obtained in the i-th step of following the option.

If o(u,v) is the preferred option in state u, then $\mathrm{V}(u) = \mathrm{Q}(u, o(u,v))$. Using the property that $0 \le r_i \le 1$,

$$\mathrm{E}_l[\gamma^l \mathrm{V}(v)] \le \mathrm{V}(u) \le \mathrm{E}_l[\gamma^l \mathrm{V}(v) + \sum_{i=1}^{l} \gamma^{i-1}]$$

$$\mathrm{E}_l[\gamma^l]\,\mathrm{V}(v) \le \mathrm{V}(u) \le \mathrm{E}_l[\gamma^l]\,\mathrm{V}(v) + \frac{1}{1-\gamma}. \qquad (1)$$

[1]This condition is equivalent to saying that the option takes the agent from u to v in finite time, and hence is not particularly strong.

E_l is an expectation over the length of the option. Using the property that $o(u,v)$ is robust, we move closer to v with probability $\bar{\epsilon} = 1-\epsilon$; this is exactly the setting of the well-studied gambler's ruin problem, where the gambler begins with a budget of $\|u-v\|$, and wins with a probability of ϵ. Using a standard result from Feller[3], with $m = \|u-v\|$, we have,

$$E_l\left[x^l\right] = \sum_{l=0}^{\infty} P\left(L=l\right) x^l \quad = \quad \frac{1}{\lambda_1^m(x) + \lambda_2^m(x)},$$

where $\lambda_1(x) = \frac{1+\sqrt{1-4\epsilon\bar{\epsilon}x^2}}{2\bar{\epsilon}x}$, and $\lambda_2(x) = \frac{1-\sqrt{1-4\bar{\epsilon}\epsilon x^2}}{2\bar{\epsilon}x}$. When $x \leq 1$,

$$\frac{1}{(\lambda_1(x)+\lambda_2(x))^m} \leq E_l[x^l] \leq \sum_{l=m}^{\infty} P(L=l)x^l$$

$$\frac{1}{(\frac{2}{2\bar{\epsilon}x})^m} \leq E_l[x^l] \leq \sum_{l=m}^{\infty} P(L=l)x^m$$

$$(\bar{\epsilon}x)^m \leq E_l[x^l] \leq x^m.$$

Substituting $x=\gamma$ and $m = \|u-v\|$ into Equation (1), we get,

$$\mathrm{E}_l[\gamma^l]\,\mathrm{V}(v) \leq \quad \mathrm{V}(u) \quad \leq \mathrm{E}_l[\gamma^l]\,\mathrm{V}(v) + \frac{1}{1-\gamma}$$

$$(\bar{\epsilon}\gamma)^{\|u-v\|}\,\mathrm{V}(v) \leq \quad \mathrm{V}(u) \quad \leq \gamma^{\|u-v\|}\,\mathrm{V}(v) + \frac{1}{1-\gamma}$$

$$\|u-v\| \log\frac{1}{\gamma} - \log\frac{1}{1-\gamma} \leq \|u-v\|_V \leq \|u-v\| \log\frac{1}{\bar{\epsilon}\gamma}.$$

□

Thus, an ϵ-greedy agent acting with respect to its value function can reach the maxima of the value function using just $O((\log|S|)^2)$ decisions. Though this result strictly applies only to the lattice setting, we observe that many MDPs are composed of lattice-like regions of local connectivity connected via bottleneck states. The presence of such bottleneck states would only increase the expected time by a constant factor.

4. OPTIONS FROM EXPERIENCE

In Section 3, we remarked that we needed $O(|S|)$ options. In order to be practical, we require an algorithm to efficiently generate these options within a budget of training epochs. The proof of Theorem 1 provides us with a crucial insight – our options only need bring the agent into an exponentially smaller *neighbourhood* of the maximal value state. This suggests that cheaply generated options may still be acceptable.

The algorithm (Algorithm 1) we propose takes a given MDP M, and trains an agent to learn T different tasks (i.e. different R) on it, evenly dividing the epoch budget amongst them. With each learned task, we certainly will have a good policy for path options from any state to the state of maximal value, M_v. However, we observe that will also have a good policy for path options from u to v is the path is 'along the gradient' of Q, i.e. when $V(u) < V(v) < V(M_v)$. Observing that $V(s) \approx \mathrm{argmax}_v\, Q(s,\pi(s))$, we detail the algorithm to construct many options options from a single Q-value function in Algorithm 2.

Algorithm 1 Small World Options from Experience

Require: M, $\mathcal{R}$, r, n, epochs, T
1: $O \leftarrow \emptyset$
2: **for** $i = 0 \rightarrow T$ **do**
3: $\quad R \sim \mathcal{R}$
4: $\quad Q \leftarrow$ Solve M with R using $\frac{\text{epochs}}{T}$ epochs
5: $\quad O' \leftarrow$ QOptions(Q, r, $\frac{n}{T}$)
6: $\quad O \leftarrow O \cup O'$
7: **end for**
8: **return** A random subset of n options from O

Algorithm 2 QOptions: Options from a Q-Value Function

Require: Q, r, n
1: $O \leftarrow \emptyset$
2: $\pi \leftarrow$ greedy policy from Q
3: **for all** s in S **do**
4: $\quad$ Choose an s' according to P_r
5: $\quad$ **if** $Q(s',\pi(s')) > Q(s,\pi(s))$ **then**
6: $\quad\quad O \leftarrow O \cup \langle\{s\},\pi,\{s'\} \cup \{t \mid Q(s',\pi(s')) < Q(t,\pi(t))\}\rangle$
7: $\quad$ **end if**
8: **end for** s in S
9: **return** A random subset of n options from O

We note here except for sampling s' from P_r, we do not require any knowledge of the MDP, nor do we need to construct a local model of the same. In fact, s' can be sampled approximately using the expected path length instead of the graph distance in P_r. As the expected path length $\mathrm{E}[l]$ is only a constant factor greater than l ($\frac{l}{\bar{\epsilon}}$), Lemma 1 continues to hold.

5. EXPERIMENTAL RESULTS

We trained MacroQ learning agents on several standard domains, and measured the cumulative return obtained using the following option generation schemes:

- **None**: No options were used.
- **Random**: Options were generated by randomly connecting two nodes in the domain (this is equivalent to P_0).
- **Betweenness**: As a representative of bottleneck-based schemes, options were generated to take any node to a local maxima of betweenness centrality, as described in [2].
- **Small World**: Options were generated randomly connecting two nodes of the domain using an inverse square law, as described in Section 3.

Each experiment, unless mentioned otherwise, was run for 10 randomly generated tasks in the domain; each task ran for $40,000$ epochs, and was averaged over an ensemble of 20 agents.

5.1 Optimal Options

The agents were run on the following three domains using the algorithm sketched in Section 3:

	Arbt. Navi	Rooms	Taxi
None	-31.82	-1.27	-16.90
Random	-31.23	-10.76	-18.83
Betw.	-18.28	-8.94	**80.48**
Sm-W	**-14.24** $[r = 4]$	**8.54**$[r = 2]$	0.66 $[r = 0.75]$

Table 1: Cumulative Return

- **Arbitrary Navigation**: The agent must reach an arbitrary goal state in an obstacle-free $x \times y$ grid-world.
- **Rooms**: The agent must navigate a floor plan with 4 rooms to reach an arbitrary goal state.
- **Taxi**: This is the domain described in Example 1.

Optimal policies were given to the options generated according to the schemes described above.

The results of these experiments are summarised in Table Table 1. Small world options perform significantly better than the other schemes in navigation-oriented tasks like Rooms or Arbitrary Navigation. In the Taxi domain, options generated by the betweenness scheme outperform the small world options. This is expected because the goal states in this domain lie at betweenness maxima.

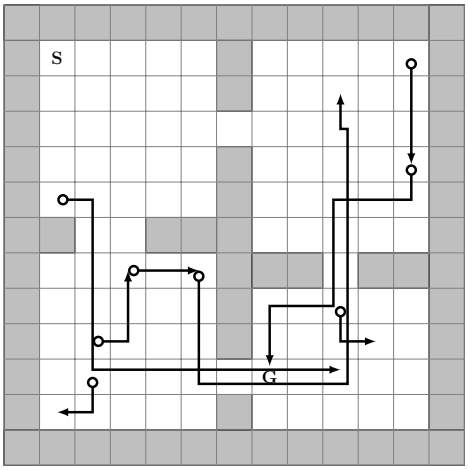


Figure 3: Rooms: Options learnt

Some of the small world options preferred in Rooms domain are shown in Figure 3. The graph shows several examples of options that compose together to arrive near the goal state. We have also plotted the learning behaviour in Figure 4. The option scheme "Betweenness + SW" combines options to betweenness maxima with small world options. Expectedly, it significantly outperforms all other schemes. The options to betweenness maxima help take the agent between strongly connected regions, while the small world options help the agent navigate within the strongly connected region.

5.2 Sensitivity of r

Figure 5 plots r versus the cumulative return on the Rooms domain. We do not yet have a clear understanding of how the exponent r should be chosen. The performance of the agent without options after $20,000$ epochs is also plotted for reference. There is a range of r (≈ 0.75 to 1.5) with good

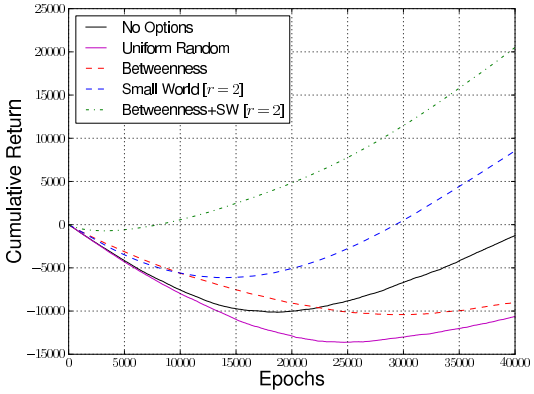


Figure 4: Rooms: Cumulative Return with 200 options

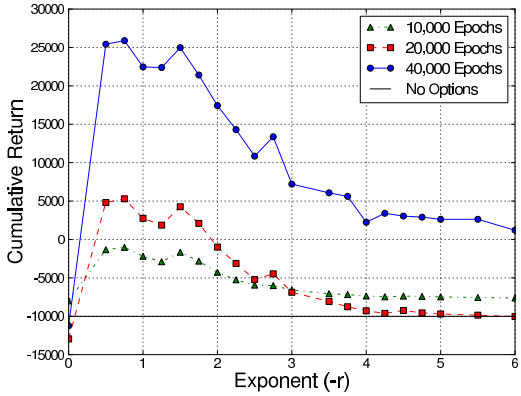


Figure 5: Rooms: r vs Cumulative Return

performance, after which the performance steadily drops. This behaviour is easily explained; as the exponent goes up, the small world options generated are very short, and do not help the agent get nearer to the maximal value state. The optimal range of r is slightly counter-intuitive because the Rooms domain is a two dimensional lattice with some edges removed. As a consequence of the reduced connectivity, and perhaps due to stochastic factors, longer range options are preferred.

5.3 Options Learnt on a Budget

In Section 4, we describe an algorithm to construct small world options efficiently when given a limited number of learning epochs. We compared the performance of these options with betweenness options learnt similarly, and have plotted our results in Figure 6. Despite using many more options, the small world options thus created significantly outperform betweenness options learnt with the same budget, and are even comparable to the optimal betweenness options.

6. CONCLUSIONS AND FUTURE WORK

We have devised a new scheme to generate options based on small world network model. The options generated sat-

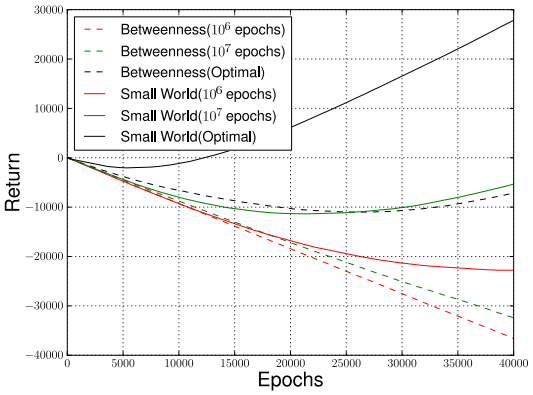


Figure 6: Rooms: Options Learnt on a Budget

isfy an intuitive criteria, that the subtasks learnt should be easily composed to solve any other task. The options greatly improve the connectivity properties of the domain, without leading to a state space blow up.

Experiments run on standard domains show significantly faster learning rates using small world options. At the same time, we have shown that learning small world options can be cheaper than learning bottleneck options, using a natural algorithm that extracts options from a handful of tasks it has solved. Another advantage of the scheme is that is does not require a model of the MDP.

As future work, we would like to characterise what the exponent r should be in a general domain. There are some technicalities to be worked out in extending our results to the continuous domain; however, as most real-life applications are continuous in nature, this is an important further direction we are looking at. Given the ease with which options can be discovered, it would be interesting to experiment with a dynamic scheme that adds options on the fly, while solving tasks. [7] extend Kleinberg's results to arbitrary graphs by using rank instead of lattice distance. It would be interesting to extend this approach to the reinforcement learning setting. The logarithmic bounds on the number of decisions presented may have some interesting consequences on theoretical guarantees of sample complexity as well.

7. REFERENCES

[1] A. G. Barto and S. Mahadevan. Recent Advances in Hierarchical Reinforcement Learning Markov and Semi-Markov Decision Processes. pages 1–28, 2003.

[2] O. Şimşek and A. G. Barto. Skill characterization based on betweenness. In *NIPS*, pages 1–8, 2008.

[3] W. Feller. *An Introduction to Probability Theory and Its Applications*, volume 1. Wiley, 1968.

[4] M. Jerrum and A. Sinclair. Conductance and the rapid mixing property for markov chains: the approximation of permanent resolved. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, STOC '88, pages 235–244, New York, NY, USA, 1988. ACM.

[5] J. Kleinberg. The Small-World Phenomenon : An Algorithmic Perspective. *ACM Theory of Computing*, 32:163–170, 2000.

[6] L. Li, T. J. Walsh, and M. L. Littman. Towards a Unified Theory of State Abstraction for MDPs. In *In Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, pages 531–539, 2006.

[7] D. Liben-Nowell, J. Novak, R. Kumar, P. Raghavan, and A. Tomkins. Geographic routing in social networks. *PNAS*, pages 1–6, 2005.

[8] C. Martel and V. Nguyen. Analyzing Kleinberg's (and other) Small-world Models. In *PODC*, volume 2, 2004.

[9] A. McGovern and A. G. Barto. Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density. In *ICML*, pages 1–8, 2001.

[10] I. Menache, S. Mannor, and N. Shimkin. Q-Cut - Dynamic Discovery of Sub-Goals in Reinforcement Learning. In *ECML*, 2002.

[11] B. Ravindran and A. G. Barto. Relativized Options : Choosing the Right Transformation. In *International Conference on Machine Learning*, 2003.

[12] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and Semi-MDPs : Learning , Planning , and Representing Knowledge at Multiple Temporal Scales at Multiple Temporal Scales. *Artificial Intelligence*, 112:181–211, 1999.

[13] A. Tahbaz-Salehi and A. Jadbabaie. Small world phenomenon, rapidly mixing markov chains, and average consensus algorithms. In *Decision and Control, 2007 46th IEEE Conference on*, pages 276 –281, 2007.

[14] M. E. Taylor and P. Stone. Transfer Learning for Reinforcement Learning Domains: A Survey. *Journal of Machine Learning Research*, 10:1633–1685, 2009.

[15] S. Thrun and A. Schwartz. Finding Structure in Reinforcement Learning. In *Advances in Neural Information Processing Systems 7*, pages 385–392, 1995.

APPENDIX

A. SMALL WORLDS

In this section we will tackle the proof of the main theorem in Section 3,

THEOREM 2. *Let $f : V \to \mathbb{R}$ be a function embedded on the graph $\mathcal{G}(V, E)$, such that, $\kappa_1\|u - v\| - c_1 \leq \|f(u) - f(v)\| \leq \kappa_2\|u-v\|-c_2$, where $0 \leq \kappa_1 \leq \kappa_2$, and $0 \leq c_2 \leq \frac{c_1}{2}$. Let M_f be the global maxima of f. Let $\mathcal{GA}_\epsilon$ be an ϵ-greedy algorithm with respect to f, i.e. an algorithm which chooses with probability $1 - \epsilon$ to transit to the neighbouring state closest to M_f, i.e. $N(u) = \text{argmin}_v \|f(v) - f(M_f)\|$.*

If $\mathcal{G}(V, E)$ is r-dimensional lattice, and contains a long distance edge distributed according to $P_r : p(u, v) \propto \|u - v\|^{-r}$, then $\mathcal{GA}_\epsilon$ takes $O((\log |V|)^2)$ steps to reach M_f.

PROOF. This result is a simple extension of Kleinberg's result in [5], and follows the proof presented there, albeit with the somewhat cleaner notation and formalism of [8]. We begin by defining the necessary formalism to present the proof.

Definition 2. Let us define $\mathrm{B}_l(u)$ to be the set of nodes contained within a "ball" of radius l centered at u, i.e. $\mathrm{B}_l(u) = \{v \mid \|u - v\| < l\}$, and $\mathrm{b}_l(u)$ to be the set of nodes on its surface, i.e. $\mathrm{b}_l(u) = \{v \mid \|u - v\| = l\}$.

Given a function $f : V \to \mathbb{R}$ embedded on $\mathcal{G}(V, E)$, we analogously define $\mathrm{B^f}_l(u) = \{v \mid |f(u) - f(v)| < l\}$. For notational convenience, we take $\mathrm{B^f}_l$ to be $\mathrm{B^f}_l(M_f)$.

The inverse normalised coefficient for $p(u, v)$ is,

$$\begin{aligned} c_u &= \sum_{v \neq u} \|u - v\|^{-r} \\ &= \sum_{j=1}^{r(n-1)} \mathrm{b}_j(u)\, j^{-r}. \end{aligned}$$

It can easily be shown that the $\mathrm{b}_l(u) = \Theta(l^{k-1})$. Thus, c_u reduces to a harmonic sum, and is hence equal to $\Theta(\log n)$. Thus, $p(u, v) = \|u - v\|^{-r}\Theta(\log n)^{-1}$.

We are now ready to prove that $\mathcal{GA}_\epsilon$ takes $O((\log |V|)^2)$ decisions. The essence of the proof is summarised in Figure 7. Let a node u be in phase j when $u \in \mathrm{B^f}_{2^{j+1}} \setminus \mathrm{B^f}_{2^j}$. The probability that phase j will end this step is equal to the probability that $N(u) \in \mathrm{B^f}_{2^j}$.

The size of $\mathrm{B^f}_{2^j}$ is at least $|\,\mathrm{B}_{\frac{2^j+c_2}{\kappa_2}}\,| = \Theta(\frac{2^j+c_2}{\kappa_2})$. The distance between u and a node in $\mathrm{B^f}_{2^j}$ is at most $\frac{2^{j+1}+c_1}{\kappa_1} + \frac{2^j+c_2}{\kappa_2} < 2(\frac{2^{j+1}+c_2}{\kappa_2})$. The probability of a link between these two nodes is at least $(\frac{2^{j+2}+2c_1}{\kappa_1})^{-r}\Theta(\log n)^{-1}$. Thus,

$$\begin{aligned} P\left(u, \mathrm{B^f}_{2^j}\right) &\geq \frac{(1-\epsilon)}{\Theta(\log n)} \left(\frac{2^j + c_2}{\kappa_2}\right)^r \times \left(\frac{2^{j+2} + 2c_1}{\kappa_1}\right)^{-r} \\ &\geq \frac{(1-\epsilon)}{\Theta(\log n)} \times \left(\frac{\kappa_1}{4\kappa_2}\right)^r \times \left(\frac{1 + \frac{c_2}{2^j}}{1 + \frac{c_1}{2\times 2^j}}\right)^r \\ &\geq \frac{(1-\epsilon)}{\Theta(\log n)} \times \left(\frac{\kappa_1}{4\kappa_2}\right)^r \times \left(\frac{1 + c_2}{1 + \frac{c_1}{2}}\right)^r. \end{aligned}$$

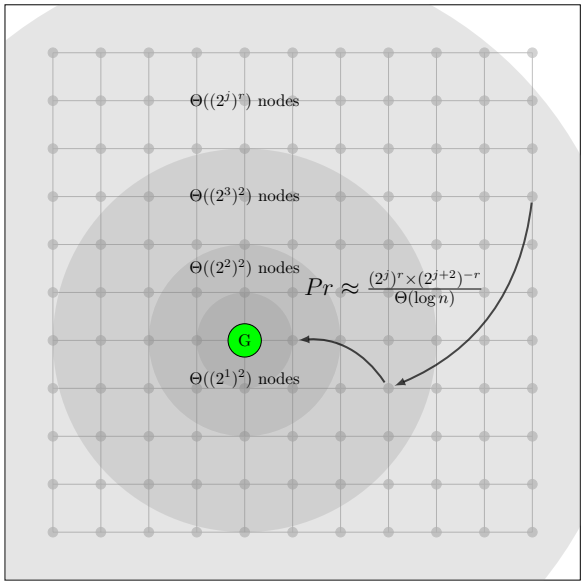


Figure 7: Exponential Neighbourhoods

Let number of decisions required to leave phase j be X_j. Then,

$$\begin{aligned} \mathrm{E}[X_j] &\leq \sum_{i=0}^{\infty} \left(1 - P\left(u, \mathrm{B^f}_{2^j}\right)\right)^i \\ &\leq \frac{1}{P\left(u, \mathrm{B^f}_{2^j}\right)} \\ &\leq \Theta(\log n) \frac{1}{(1-\epsilon)} \left(\frac{4\kappa_2}{\kappa_1}\right)^r \left(\frac{1 + \frac{c_1}{2}}{1 + c_2}\right)^r \\ &\leq \Theta(\log n). \end{aligned}$$

Thus, it takes at most $O(\log n)$ decisions to leave phase j. By construction, there are at most $\log n$ phases, and thus at most $O((\log n)^2)$ decisions. □

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

Jacob W. Crandall
Computing and Information Science Program
Masdar Institute of Science and Technology
Abu Dhabi, United Arab Emirates
jcrandall@masdar.ac.ae

ABSTRACT

Learning in multi-agent settings has recently garnered much interest, the result of which has been the development of somewhat effective multi-agent learning (MAL) algorithms for repeated normal-form games. However, general-purpose MAL algorithms for richer environments, such as general-sum repeated stochastic (Markov) games (RSGs), are less advanced. Indeed, previously created MAL algorithms for RSGs are typically successful only when the behavior of associates meets specific game theoretic assumptions and when the game is of a particular class (such as zero-sum games). In this paper, we present a new algorithm, called Pepper, that can be used to extend MAL algorithms designed for repeated normal-form games to RSGs. We demonstrate that Pepper creates a family of new algorithms, each of whose asymptotic performance in RSGs is reminiscent of its asymptotic performance in related repeated normal-form games. We also show that some algorithms formed with Pepper outperform existing algorithms in an interesting RSG.

Categories and Subject Descriptors

I.2.6 [**Computing Methodologies**]: Artificial Intelligence-Learning

General Terms

Algorithms

Keywords

Multi-agent learning, stochastic games, game theory

1. INTRODUCTION

Much research in multi-agent learning (MAL) continues to focus on learning in repeated normal-form (or *matrix*) games. This research has resulted in many *matrix learning algorithms* (MLAs – algorithms for learning repeated matrix games), some of which are able to learn effectively in general-sum matrix games. For example, M-Qubed has been shown to perform robustly in several empirical studies involving many different MLAs in many repeated matrix games [4, 10].

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

Exp3 [2], GIGA-WoLF [6], and UCB [1] have also performed well in certain games in these studies.

However, many situations in which MAL is necessary are better modeled as repeated stochastic games (RSGs) than repeated matrix games. In RSGs, rewards are received incrementally. This requires a learning agent to solve two problems simultaneously. First, it must determine its delayed rewards, or the rewards it will receive in future states of the world. These delayed rewards are contingent on the strategies of the agent and its associates in future states. Second, the agent (and its associates) must learn an effective strategy in each state. But learning such strategies requires accurate estimates of delayed rewards.

Whereas these chicken and egg problems can be solved simultaneously in single-agent domains via standard reinforcement learning, they are not so easily solved in multi-agent settings since an agent cannot easily predict or control its associates' current and future strategies. As a result, many MAL algorithms have been investigated for RSGs. These algorithms typically assume that associates will conform to a particular game-theoretic behavior, and that the corresponding game theoretic solution concept will define an effective strategy for the agent. While sometimes effective, such algorithms often do not perform well when these assumptions fail. For example, algorithms that seek to learn a Nash equilibrium (NE) sometimes perform well in competitive stochastic games in which the NE corresponds to the minimax strategy (e.g., [17]), but often fail in repeated general-sum stochastic games when there is not a unique NE [5] (see also the folk theorem [12]). Additionally, many algorithms require perfect knowledge of associates' payoffs, which are often unavailable.

In this paper, we introduce a new algorithm, called *Pepper*, for learning in RSGs for situations in which the actions of associates are observable, but where the payoffs of associates and (initially) the state transitions of the game are unknown. This new algorithm uses a separate instance of a MLA to learn a strategy in each unique state of the game using payoffs (future rewards) computed by Pepper. To demonstrate the effectiveness of Pepper, we use it to derive four new algorithms for RSGs. We then evaluate these algorithms in an interesting RSG played against a variety of learning algorithms and hand-coded (static) strategies.

2. BACKGROUND AND AN EXAMPLE

We begin by defining important terms and notation. We then review the performance of existing algorithms in an example RSG.

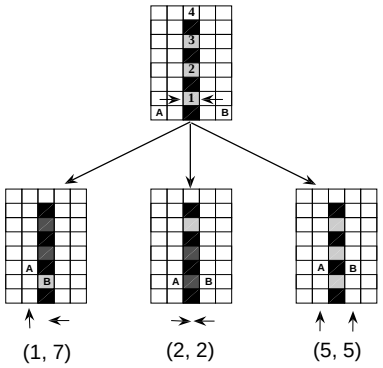


Figure 1: The SGPD game.

2.1 Definitions and Notation

We consider repeated stochastic games (RSGs) played by players[1] (or agents) i and $-i$. An RSG consists of a set of *stage games* S, also known as world states. In each stage $s \in S$, both players choose an action from a finite set. Let $A(s) = A_i(s) \times A_{-i}(s)$ be the set of joint actions available in s, where $A_i(s)$ and $A_{-i}(s)$ are the action sets of players i and $-i$, respectively. Each *episode* of the game begins in some beginning stage $s_b \in B \subseteq S$ and terminates when some goal stage $s_g \in G \subseteq S$ is reached. Once a goal stage is encountered, a new episode begins in some stage $s_b \in B$.

When joint action $\mathbf{a} = (a_i, a_{-i})$ is played in stage s, player i receives a finite reward $r_i(s, \mathbf{a})$. Here, $a_i \in A_i(s)$ and $a_{-i} \in A_{-i}(s)$ are the actions of players i and $-i$, respectively. Once the joint action $\mathbf{a}$ is played in s, the world transitions to some new stage game s' according to the probabilistic stage transition model $P_M(s, s', \mathbf{a})$.

We assume that the transition model P_M is unknown to the players initially. However, we assume that the players can learn the transition model from experience by observing joint actions and stage transitions. Additionally, we assume that players can observe their own immediate rewards, but not those of associates. We also assume that player i's maximum possible reward R_i^{max} for an episode is known *a priori*. This assumption can be replaced by a process through which R_i^{max} is learned, but this is not essential to our discussion.

2.2 Motivating Example

Figure 1 depicts a stochastic game prisoner's dilemma (SGPD) [13]. In this game, two players, labeled `A` and `B`, begin each episode in opposite corners of the world. The goal of the game is to enter one of the gates (labeled 1, 2, 3, and 4 in the figure) in as few moves as possible. If both agents try to enter gate 1 at the same time, then gates 1 and 2 close and the agents must enter gate 3. If only one agent enters gate 1, gates 1–3 close and the other agent must enter gate 4. If either agent enters gate 2, then gate 1 closes. If both agents try to enter gate 2 they are both allowed passage. An episode ends once both agents have entered a gate.

The set of stages of the SGPD is defined by the positions of the players and the status of the four gates. Each configuration of these elements is a unique stage game in which both players' can choose to move *up*, *down*, or *toward the gates* (*right* for player `A`, and *left* for player `B`). Moves into a wall (boundary, black space, or closed gate) result in no movement. Each player receives 10 points for entering any gate, and is penalized 1 point for each move it takes.

[1]For ease of exposition, we assume the game has two players; Pepper can easily be extended to $(n > 2)$-player games.

Table 1: High-level payoff matrix of the SGPD.

	Defect (Gate 1)	**Cooperate** (Gates 2–4)
Defect (Gate 1)	2, 2	7, 1
Cooperate (Gates 2–4)	1, 7	5, 5

Table 2: Performance in the SGPD. Bold indicates nearly ideal performance. Algorithms marked *VI* use model-based, rather than model free, methods.

Algorithm	**Associate**				
	Self Play	**Defector**	**Cooperator**	**Random**	**TFT**
Q-learning	2.2	1.3	**7.0**	**4.3**	3.4
WoLF-PHC	1.9	1.2	**6.9**	**4.1**	2.1
Minimax-VI	2.0	**1.9**	**7.0**	**4.4**	2.0
Friend-VI	1.9	-0.9	3.0	1.0	2.0
Nash-VI*	2	**2**	**7**	**4.5**	2
uCE-VI*	**5**	1	5	3	**5**
FolkEgal*†	**5**	**2**	5	3.75	**5**

** Requires knowledge of associate's payoffs; performance estimated
† Requires knowledge of stage transition probabilities $P_M(s, s', \mathbf{a})$

A player that tries to enter gate 1 is said to have *defected*. Otherwise, the player is said to have *cooperated*. Thus, the *high-level* game is the prisoner's dilemma (PD) matrix game shown in Table 1; each cell lists the sum of rewards received in an episode by the row and column players, respectively.

Table 2 shows the average asymptotic per episode payoffs of existing algorithms in the SGPD in self play and when associating with four static associates. Defector always defects, Cooperator always cooperates, Random chooses a gate randomly, and TFT chooses a gate according to the tit-for-tat strategy. Bold values indicate (nearly) ideal performance. The payoffs of Q-learning [24], WoLF-PHC [7], Minimax-VI [19], and Friend-VI [20] are based on 200,000-episode games run using the parameter settings given in Table 5. The payoffs of Nash-VI [16], utilitarian correlated reinforcement learning (uCE-VI) [14], and FolkEgal [9] were deduced from the descriptions of the algorithms.

Table 2 shows that none of the algorithms performs ideally against this limited set of algorithms. Only uCE and FolkEgal cooperate in self play, which results in an expected per-episode payoff of 5. However, uCE and FolkEgal do not behave ideally against Cooperator or Random. Additionally, they must know their associate's payoffs.

We seek to identify MAL algorithms that learn effectively against many kinds of associates in the SGPD and other general-sum RSGs when associate's payoffs are unknown. Since several matrix learning algorithms (MLAs) have, to various degrees, achieved these objectives in matrix games (such as Table 1), we explore the extension of these algorithms to RSGs using Pepper.

3. PEPPER

Pepper (*p*otential *exp*loration with *ps*eudo stationary *r*estarts) is defined by three design choices. First, Pepper utilizes the R-max [8] algorithmic framework, which incorporates the *optimism-in-uncertainty* principle. Second, Pep-

per defines a new mechanism for estimating future rewards, which are used to estimate a payoff matrix for each stage of the RSG. This mechanism is also designed to adhere to the optimism-in-uncertainty principle, while eventually reflecting the agent's actual rewards in an episode. Third, Pepper uses a separate instance of an MLA in each stage of the game to learn the agent's strategy in that stage. The MLA employed in stage $s \in S$ learns the agent's policy in s as it would learn a policy in a repeated matrix game, only it learns from a payoff matrix defined by Pepper.

3.1 Algorithmic Framework

To determine how to act in stage s, player i must determine how its actions will affect it rewards in the remainder of the episode (i.e., its future rewards). To do this, Pepper computes the payoff matrix $R_i(s)$, which estimates player i's future rewards for each joint action $\mathbf{a} \in A(s)$. Let $R_i(s, \mathbf{a})$ denote the expected future rewards obtained once the joint action $\mathbf{a}$ is played in stage s. Also, let $s_k \in S$ be the k^{th} stage visited in an episode in which T stages are visited. Then, $R_i(s, \mathbf{a})$ is given by

$$R_i(s_k, \mathbf{a}) = \sum_{\tau=k}^{T} r_i^t(s_\tau, \mathbf{a}_\tau), \tag{1}$$

where $\mathbf{a}_\tau$ is the joint action taken in stage s_τ. Bellman [3] showed that $R_i(s_k, \mathbf{a})$ can be equivalently expressed as

$$R_i(s_k, \mathbf{a}) = r_i(s_j, \mathbf{a}) + \sum_{s' \in S} P_M(s_k, s', \mathbf{a})\ V_i(s'), \tag{2}$$

where $V_i(s')$ is the expected future rewards for being in stage s'. Thus, given $r_i(s, \mathbf{a})$, $P_M(s, s', \mathbf{a})$, and $V_i(s')$, $R_i(s, \mathbf{a})$, for each $s \in S$, can be computed using value iteration.

When $r_i(s, \mathbf{a})$, $P_M(s, s', \mathbf{a})$ and $V_i(s')$ are unknown, an agent must learn them via exploration. In constant-sum RSGs, the R-max algorithm [8] does this using the optimism-in-uncertainty principle. R-max initially assumes that each joint action from each stage results in maximal reward. It removes this assumption once the joint action has been attempted K (predetermined) times. This draws the agent toward stages that have not been adequately explored.

Algorithm 1 embodies principles of the R-max algorithm for RSGs minus rules specifying how $\pi_i(s)$ or $V_i(s)$ are computed. This algorithmic framework can be used to implement many MAL algorithms, including R-max and the algorithms listed in Table 3, each of which computes $\pi_i(s)$ and $V_i(s)$ differently. The algorithms in Table 3 each assume that all players will collectively conform to some game theoretic behavior, such as minimax (Minimax-Q), NE (Nash-Q), correlated equilibria (CE-Q), or the Nash bargaining solution (NBS-Q) [21]. When appropriate assumptions are met, $R_i(s, \mathbf{a})$ will converge to its "true" value, as it does in single-agent reinforcement learning. However, when these assumptions are not met as is often the case, these algorithms often achieve low payoffs.

As an alternative, Pepper proposes a method for creating a new family of algorithms for RSGs. This method also utilizes Algorithm 1, but it proposes a new mechanism for how $\pi_i(s)$ and $V_i(s)$ should be computed.

3.2 Computing a Strategy $(\pi_i(s))$

There exist MLAs that define strategy selection rules that are successful in many repeated matrix games played with many different associates. Pepper seeks to leverage these learning rules by extending MLAs to RSGs. In particular, Pepper uses an MLA to learn a strategy $\pi_i(s)$ in each stage $s \in S$. The MLA used in stage s seeks to learn a strategy that, given the strategy played by its associate in s, maximizes the payoffs defined by the payoff matrix $R_i(s)$.

Given that the MLA in each stage s seeks to maximize the agent's payoffs as defined by $R_i(s)$, the ability of the MLA employed in stage $s \in S$ to learn a successful strategy is contingent on accurate estimates of $R_i(s)$. Accurately modeling $R_i(s)$ depends, in turn, on effective estimates of future reward, which are encoded by $V_i(s')$.

Algorithm 1 Algorithmic framework

Input:
Let $S' = \{s_0, S\}$ be a set of stage games, where $s_0 \in G$
Let R_i^{max} be player i's maximum reward for an episode
Initialize:
$\forall s \in S', \mathbf{a} \in A(s), r_i(s, \mathbf{a}) = 0$, $P_M(s, s_0, \mathbf{a}) = 1$, $\kappa(s, \mathbf{a}) \leftarrow 0$
$\forall s \in S', V_i(s) = R_i^{max}$ and $\pi_i(s) =$ random
$t \leftarrow 1$
repeat
$\forall s \in S, \mathbf{a} \in A(s)$, update $R_i(s, \mathbf{a})$ by value iteration; Eq. (2)
$\forall s \in S$, update $\pi_i(s)$ and $V_i(s)$
Observe starting state $s \in B$
repeat
Select action a_i according to $\pi_i(s)$ and execute
Observe $\mathbf{a}^t = (a_i, a_{-i})$, s', and r_i
$\kappa(s, \mathbf{a}) \leftarrow \kappa(s, \mathbf{a}) + 1$
if $\kappa(s, \mathbf{a}) \geq K$ **then**
Update $r_i(s, \mathbf{a})$ according to observations
Update $V_i(s)$
Update $P_M(s, \cdot, \mathbf{a})$ according to observed frequencies
Update $R_i(s, \mathbf{a})$ according to Eq. (2)
Update $\pi_i(s)$
end if
$s \leftarrow s'$
until $s \in G$
until Game Over

Table 3: Algorithms generalized by Algorithm 1.

Algorithm	Computing $\pi(s)$	Computing $V_i(s')$
Minimax-Q	maximin strat. of $R_i(s)$	maximin value of $R_i(s)$
Nash-Q	A NE of $R_i(s)$	value of a NE of $R(s)$
Friend-Q	$\arg\max_{\mathbf{a} \in A(s)} R_i(s, \mathbf{a})$	$\max_{\mathbf{a} \in A(s)} R_i(s, \mathbf{a})$
CE-Q	A CE of $R_i(s)$	value of a CE of $R(s)$
NBS-Q	NBS of $R(s)$	NBS value of $R(s)$

3.3 Determining Future Rewards $(V_i(s))$

We advocate that estimates of $V_i(s)$ should satisfy two objectives, or properties:

Realism Property: $V_i(s)$ must eventually reflect the actual payoffs received by the agent in an episode after stage s is reached. That is, $\lim_{t \to \infty} V_i^t(s) = \bar{V}_i(s)$, where $V_i^t(s)$ is the estimate of the expected rewards received in episode t after stage s is reached, and $\bar{V}_i(s)$ is the actual expected sum of rewards received in an episode after s is reached.

Optimism Property: $V_i(s)$ should overestimate, rather than underestimate, the agent's future reward while learning. That is, for all t, $V_i^t(s) \geq \bar{V}_i(s)$.

The realism property ensures that the MLA employed in stage s eventually learns from true expected payoffs, while the optimism property, similar to an *admissible heuristic* in

search, helps the agent to avoid learning strategies that lead to premature convergence to local (but not global) maxima.

Pepper seeks to satisfy these two properties by combining off-policy and on-policy methods for estimating $V_i(s)$. Off-policy methods estimate $V_i(s)$ using some ideal (often derived from $R_i(s)$) that the agent hopes to eventually reach. This ideal is specific to the MLA that is used. For example, each algorithm in Table 3 employs a different off-policy method for estimating $V_i(s)$. Let $V_i^{\text{off}}(s)$ denote the estimate of $V_i(s)$ computed from the designated off-policy method.

Alternately, on-policy methods estimate $V_i(s)$ from the actual distribution over joint actions induced by the players' joint strategy. Let $V_i^{\text{on}}(s)$ denote this estimate. Formally,

$$V_i^{\text{on}}(s) = \sum_{\mathbf{a}=(a_i,a_{-i})\in A(s)} \pi_i(s,a_i)\ \pi_{-i}(s,a_{-i})\ R_i(s,\mathbf{a}), \quad (3)$$

where $\pi_i(s,a_i)$ and $\pi_{-i}(s,a_{-i})$ are the probabilities that players i and $-i$ play actions a_i and a_{-i}, respectively, in stage s. However, since player i does not know $\pi_{-i}(s)$ and since its own strategy varies over time, $V_i^{\text{on}}(s)$ can be defined in terms of the observed distribution of joint actions. Let $\kappa(s,\mathbf{a})$ be the number of times that joint action $\mathbf{a}$ has been played in stage s, and let $\kappa^t(s)$ be the number of times that stage s has been visited. Then,

$$V_i^{\text{on}}(s) = \sum_{\mathbf{a}\in A(s)} \frac{\kappa(s,\mathbf{a})}{\kappa(s)}\ R_i(s,\mathbf{a}). \quad (4)$$

In practice, $\frac{\kappa(s,\mathbf{a})}{\kappa(s)}$ can be replaced with a probability that places higher weight on more recent observations.

$V_i^{\text{on}}(s)$, as computed in Eq. (4), satisfies the realism property when $R_i(s,\mathbf{a})$ approaches true expected payoffs for each joint action $\mathbf{a}\in A(s)$. Given that $R_i(s)$ converges with high probability when each joint action in each stage is played sufficiently [8], $V_i(s)$ will converge to the average rewards received from stage s, since joint actions not played sufficiently will not substantially impact it.

However, $V_i^{\text{on}}(s)$ may not satisfy the optimism property. Rather, an agent hoping to compute $V_i(s)$ to satisfy the optimism property could estimate $V_i(s)$ as the maximum of $V_i^{\text{on}}(s)$ and $V_i^{\text{off}}(s)$, particularly since both $V_i^{\text{on}}(s)$ and $V_i^{\text{off}}(s)$ are based on initially optimistic assessments of $R_i(s)$. We denote $\hat{V}_i(s)$ as this optimistic assessment, given by

$$\hat{V}_i(s) = \max\left(V_i^{\text{off}}(s), V_i^{\text{on}}(s)\right). \quad (5)$$

Thus, $\hat{V}_i(s)$ has the best chance of satisfying the optimism property, and $V_i^{\text{on}}(s)$ satisfies the realism property. Pepper seeks to obtain the best of both worlds by computing $V_i(s)$ as a convex combination of $\hat{V}_i(s)$ and $V_i^{\text{on}}(s)$. That is,

$$V_i(s) = \lambda_i(s)\ \hat{V}_i(s) + (1-\lambda_i(s))\ V_i^{\text{on}}(s) \quad (6)$$

where $\lambda_i(s)\in[0,1]$ is set to one initially, but approaches zero as the agent obtains more experience in stage s. However, it is not clear how quickly $\lambda_i(s)$ should be decreased. If $\lambda_i(s)$ decreases too quickly, then Eq. (6) is unlikely to satisfy the optimism property. On the other hand, if $\lambda_i(s)$ decreases too slowly, then the algorithm will learn too slowly.

In attempt to avoid either extreme, Pepper regulates $\lambda_i(s)$ using a concept that we refer to as *pseudo-stationarity*. We say that payoff matrix $R_i(s)$ is pseudo-stationary if each entry of $R_i(s)$ has stopped decreasing.

Pseudo-Stationary: Let $\underline{R}_i^T(s,\mathbf{a})$ be the lowest estimate of $R_i(s,\mathbf{a})$ observed up to time T, and let $R_i^t(s,\mathbf{a})$ be the estimate of $R_i(s,\mathbf{a})$ at time t. $R_i(s)$ is *pseudo-stationary* after time T if $\forall t\geq T, \mathbf{a}\in A(s),\ R_i^t(s,\mathbf{a})\geq \underline{R}_i^T(s,\mathbf{a})+\delta$, where $\delta>0$ is some small positive constant.

Since $R_i(s,\mathbf{a})$ is initially set to R_i^{max}, it will likely decrease in early episodes. Thus, $R_i(s)$ will not likely be pseudo-stationary in early episodes of the game, but will eventually become pseudo-stationary given episodes of finite length.

Pepper uses the concept of *non-pseudo-stationary re-starts* to regulate $\lambda_i(s)$. That is, when $R_i(s)$ is observed to not be pseudo-stationary, $\lambda_i(s)$ is reset to one. This restarts the transition from $\hat{V}_i(s)$ to $V_i^{\text{on}}(s)$ defined by Eq. (6). Let $\kappa'(s)$ be the number of visits to stage s since $R_i(s)$ was last observed to not be pseudo-stationary. Then, $\lambda_i(s)$ is given by:

$$\lambda_i(s) = \max\left(0,\ \frac{C-\kappa'(s)}{C}\right), \quad (7)$$

where C is some positive integer. $\lambda_i(s)$ decreases more slowly for higher values of C than for lower values of C in the absence of non-pseudo-stationary restarts.

Since the number of restarts in each stage is finite given bounded rewards, $V_i(s)$ as defined by Eqs. (6) and (7) will converge to actual rewards if each $(s,\mathbf{a})$-pair is visited sufficiently. Since Algorithm 1 ensures with high probability that all $(s,\mathbf{a})$-pairs will be explored sufficiently given that the optimism property is met [8], $V_i(s)$ will converge to actual rewards in stages encountered in the learned solution when the optimism property is met.

4. ALGORITHMS FORMED BY PEPPER

We now describe four new algorithms for RSGs formed by extending four different MLAs with Pepper. Algorithm 1 paired with Eq. (6) requires that we must only define how $\pi_i(s)$ and $V_i^{\text{off}}(s)$ are computed. We refer the reader to the literature for specifications of how $\pi_i(s)$ is computed by each MLA, as Pepper uses these rules as they have been defined. We now specify how each algorithm computes $V_i^{\text{off}}(s)$.

4.1 M-Qubed with Pepper

M-Qubed [10] is a reinforcement learning algorithm that balances cautious, optimistic, and best-response attitudes. It encodes the previous ω joint actions taken by the agents as state (called *recurrent state*). It then learns a Q-value for each recurrent state-action pair, each of which is initialized to its highest possible value given its discount factor γ. M-Qubed typically selects actions based on its Q-values in the current (recurrent) state, but triggers to its maximin strategy when its total loss exceeds a pre-determine threshold.

M-Qubed learns to play the Nash bargaining solution in self play in many repeated matrix games. It also avoids being exploited, meaning that its long-term payoffs meet or exceed its maximin value regardless of the behavior of its associates. In the PD matrix game, it learns to cooperate in self play and to defect against associates that defect.

M-Qubed's mechanics define a relaxation search for a strategy that sustains a future discounted reward $\frac{r}{1-\gamma}$ that meets or exceed its highest current Q-value over all of its recurrent states (denoted $Q^*(s)$). Thus, basing $V_i^{\text{off}}(s)$ on $Q^*(s)$ is a natural choice. Formally, let $\Omega(s)$ be the set of M-Qubed's recurrent states in stage s, and let $\Omega'(s)\subseteq\Omega(s)$ be the set of recurrent states visited in the last τ visits to s. Also, let

$Q(\sigma, a_i)$ be the Q-value for taking action a_i in $\sigma \in \Omega$. Then,

$$V_i^{\text{off}}(s) = (1-\gamma) \cdot \left(\max_{\sigma \in \Omega'(s), a_i \in A_i(s)} Q(\sigma, a_i) \right) \quad (8)$$

In RSGs, M-Qubed's recurrent state $\sigma \in \Omega(s)$ in stage s is determined by the previous ω joint actions taken in s.

4.2 Salt and Pepper

The satisficing learning technique (Salt) is a simple MLA proposed by Karandikar et al. [18]. We use the version of the algorithm defined and analyzed by Stimpson and Goodrich [22]. Salt converges with high probability in self play to pareto efficient solutions. In the repeated PD matrix game, it learns with high probability to cooperate in self play and to defect against agents that *always* defect [22].

Salt and Pepper encodes an aspiration level $\alpha_i(s)$ in each stage $s \in S$, which is initialized to $\max_{\mathbf{a}} R_i(s, \mathbf{a})$. $\alpha_i(s)$ is then incremented toward $R_i(s, \mathbf{a})$ when $\mathbf{a}$ is played in s. When $\alpha_i^t \geq R_i(s, \mathbf{a})$, Salt and Pepper repeats its action the next time s is visited. Otherwise, it randomly selects a new action. Salt and Pepper sets $V_i^{\text{off}}(s)$ to α_i^t, which typically provides an optimistic estimate of $V_i(s)$ in early episodes.

4.3 Fictitious Play with Pepper

Fictitious play (FP) [11] is one of the oldest MLAs. It forms a simple model of its opponent by observing the empirical distribution of its opponent's actions. In each time step, it selects the action that maximizes its expected payoff given this opponent model and its payoff matrix. Formally, let $\gamma(a_{-i})$ be the percentage of time that its opponent (player $-i$) has played action a_{-i} in the past. Then, FP's utility for playing action a_i is:

$$u_i(a_i) = \sum_{a_{-i} \in A_{-i}(s)} \gamma(a_{-i})\ R_i(s, (a_i, a_{-i})). \quad (9)$$

We implemented weighted FP, in which the assessment $\gamma(a_{-i})$ gives more weight to recent observations,

FP converges to a NE in self play in matrix games that are iterative dominance solvable. In the PD matrix game, it learns to defect against all associates regardless of their propensity to cooperate or retaliate.

FP with Pepper sets $V_i^{\text{off}}(s)$ to its max utility. Formally,

$$V_i^{\text{off}}(s) = \max_{a_i \in A_i(s)} u_i(a_i). \quad (10)$$

This valuation is not optimistic when $R_i(s)$ has converged to actual payoffs. However, since $R_i(s)$ is initialized optimistically, it is optimistic initially.

4.4 GIGA-WoLF with Pepper

GIGA-WoLF [6] is a gradient ascent MLA that uses multiple learning rates to achieve no regret. Unlike the other three learning algorithms, GIGA-WoLF selects a strategy from the mixed strategy space. In the PD matrix game, GIGA-WoLF quickly learns to defect against all associates.

GIGA-WoLF with Pepper sets $V_i^{\text{off}}(s)$ to its weighted average reward (given by $R_i(s)$); newer samples receive more weight. Thus, like FP with Pepper, this valuation is optimistic initially, and falls to true values as $R_i(s)$ converges.

5. RESULTS

In this section, we evaluate the behavior and performance of M-Qubed with Pepper, Salt and Pepper, FP with Pepper, and GIGA-WoLF with Pepper in the SGPD. First, we analyze their behavior in self play and against WoLF-PHC [7], an MAL algorithm that typically learns to defect in this game. We then provide a more in-depth analysis of the robustness of these algorithms when associating with many different kinds of associates in the SGPD, including association among the various Pepper algorithms, other existing learning algorithms, and hand-coded (static) strategies. We do so by conducting a round-robin tournament and an evolutionary tournament, each involving the same 12 algorithms. Parameter values for all algorithms are provided in Table 5.

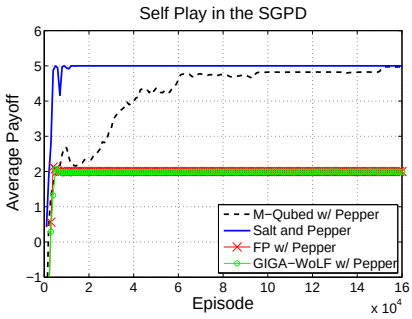


Figure 2: Average payoffs in self play over 20 trials.

5.1 Pepper in the SGPD

Figure 2 shows the average performance over time of M-Qubed with Pepper, Salt and Pepper, FP with Pepper, and GIGA-WoLF with Pepper in the SGPD in self play. The figure shows that both M-Qubed with Pepper and Salt and Pepper learn to cooperate in self play. This results in an average asymptotic payoff of 5 points per episode. However, while both of these algorithms learn to cooperate in self play, Salt and Pepper converges much faster than M-Qubed with Pepper. These results are consistent with the behavior of these algorithm in the PD matrix game, in which both Salt and M-Qubed learn to cooperate in self play, with Salt converging faster than M-Qubed.

Alternately, Figure 2 shows that both FP with Pepper and GIGA-WoLF with Pepper quickly learn to defect in self play in the SGPD. This results in an asymptotic payoff of 2 points per episode. This is substantially less than if both agents had learned to cooperate, but it is consistent with how GIGA-WoLF and FP perform in the PD matrix game.

Against WoLF-PHC in the SGPD, the average asymptotic payoff of each algorithm is near 2 points per episode (Figure 3). All four algorithms learn to defect against WoLF-PHC, though Salt and Pepper's average payoffs are slightly lower than that of mutual defection. This degraded performance is due to occasional exploration by WoLF-PHC, which causes Salt and Pepper to become dissatisfied in some stages. This requires it to re-learn how to defect, which typically takes several episodes. Again, M-Qubed with Pepper learns much slower than the other algorithms.

Despite M-Qubed with Pepper's slow learning rate, we are not aware of another learning algorithm from the literature that, without knowing its associate's payoffs, learns to both cooperate in self play and to always defect against WoLF-PHC in the SGPD. This demonstrates the effectiveness of Pepper for creating algorithms that outperform existing MAL algorithms in RSGs.

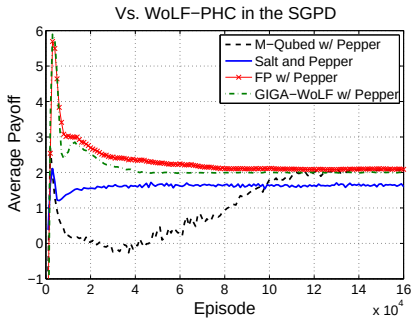


Figure 3: Average payoffs against WoLF-PHC over 20 trials.

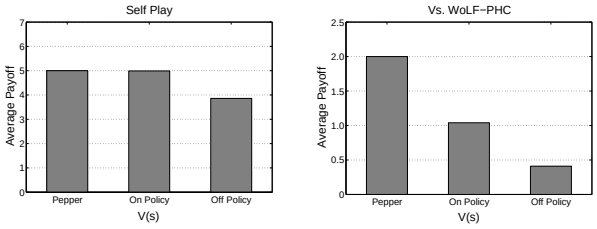
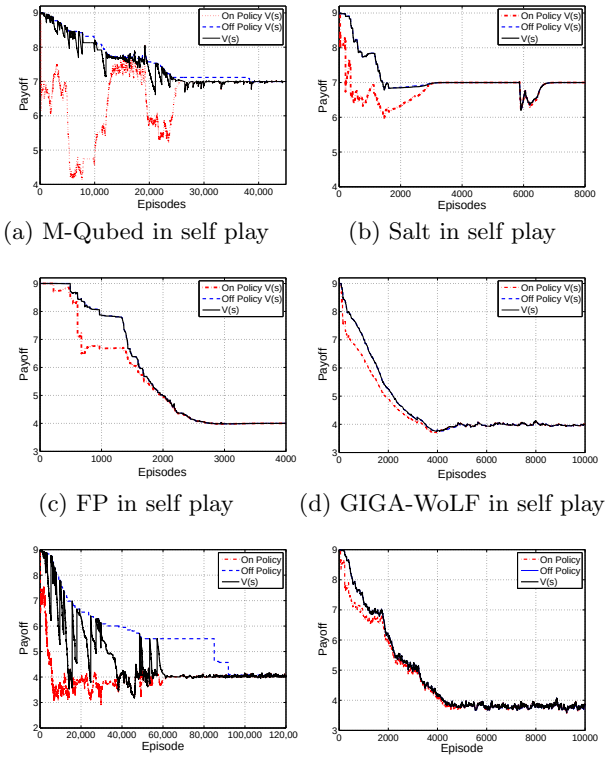


Figure 4: Average asymptotic payoffs of M-Qubed with Pepper given different estimates of $V_i(s)$ in self play and against WoLF-PHC.

The success of M-Qubed with Pepper can be traced in large part to how Pepper estimates $V_i(s)$. Figure 4 shows the average payoffs of M-Qubed in self play and against WoLF-PHC given different methods for estimating $V_i(s)$. When $V_i(s)$ is set equal to $V_i^{\text{on}}(s)$ (On Policy), M-Qubed still learns to cooperate in self play, but it gets exploited by WoLF-PHC. When $V_i(s)$ is set equal to $V_i^{\text{off}}(s)$ (Off Policy), its payoffs in self play and against WoLF-PHC are substantially lower than when $V_i(s)$ is set by Pepper. In fact, in one trial (not reflected in Figure 4) in self play, using the off-policy valuation caused both agents to converge to a strategy in which neither agent entered a gate within 200 moves.

To better understand the $V_i(s)$ as it is computed by Pepper, consider Figure 5. This figure shows values of $V_i^{\text{off}}(s)$, $V_i^{\text{on}}(s)$, and $V_i(s)$ over time in the stage game in which each agent is immediately next to an open gate 1. Figures 5(a)–5(d) correspond to valuations made by each of the Pepper algorithms in self play, while Figures 5(e) and 5(f) show valuations of M-Qubed with Pepper and Salt and Pepper against WoLF-PHC. We note that, from this particular stage, mutual defection gives each agent a future reward of 4, and mutual cooperation gives each agent a future reward of 7.

We make several observations about Figure 5. First, all valuations eventually converge to the true value of the stage in question in each scenario. Second, $V_i^{\text{off}}(s)$ is often greater than $V_i^{\text{on}}(s)$ in each scenario. Thus, $V_i(s)$ tends to mirror $V_i^{\text{off}}(s)$ except in the case of M-Qubed with Pepper (Figures 5(a) and 5(e)), particularly against GIGA-WoLF. In this latter scenario, $V_i(s)$ eventually mirrors $V_i^{\text{on}}(s)$, which allows it to defect against WoLF-PHC. Third, $V_i(s)$ typically, but not always, conforms with the optimism property; $V_i(s)$ is usually greater than or equal to the eventual value of the stage. This causes the algorithms to effectively explore using the optimism-in-uncertainty principle.



(a) M-Qubed in self play (b) Salt in self play

(c) FP in self play (d) GIGA-WoLF in self play

(e) M-Qubed vs. WoLF-PHC (f) Salt vs. WoLF-PHC

Figure 5: $V_i^{\text{on}}(s)$, $V_i^{\text{off}}(s)$, and $V_i(s)$ in the SGPD for the stage where both agents are in front of gate 1.

5.2 Tournaments in the SGPD

We conducted a round-robin tournament involving 12 algorithms: the four Pepper algorithms, four other MAL algorithms (Q-learning [24], WoLF-PHC, Friend-VI [20], and Minimax-VI [20]), and four static strategies (Tit-for-Tat, Always Defect, Always Cooperate, and Random). In this tournament, each algorithm was paired with itself and the other algorithms in a 200,000-episode SGPD. Since we are primarily concerned with asymptotic performance in this paper, the performance of the algorithms was taken only in the last 10,000 episodes. Alternate evaluation windows (such as the average of all episodes) could yield different results.

The results of the round-robin tournament are shown Table 4. M-Qubed with Pepper had the highest average performance, followed by Always Defect, Tit-for-Tat, and Salt and Pepper. FP with Pepper and GIGA-WoLF with Pepper placed fifth and sixth, respectively.

In addition to learning to cooperate in self play, M-Qubed with Pepper learns mutual cooperation with Salt and Pepper. On the other hand, it learns to always defect against each of the other algorithms except Tit-for-Tat. This allows it to avoid being exploited by algorithms that are apt to defect, and to exploit algorithms that will cooperate (Always Cooperate, Friend-VI, and, to a lesser degree, Random).

However, M-Qubed with Pepper does not learn to always cooperate with Tit-for-Tat, nor do any of the other learning algorithms. While an ideal algorithm cooperates with Tit-for-Tat, the version of Tit-for-tat we implemented for the SGPD responds to the global behavior of its associate,

Table 4: Average asymptotic payoffs in the SGPD for each pairing. All results are an average of 20 trials.

Algorithm	Associate												
	M-Qubed w/ Pepper	Always Defect	TFT	Salt and Pepper	FP w/ Pepper	GIGA-WoLF w/ Pepper	Q-learning	Minimax-VI	Random	WoLF-PHC	Always Cooperate	Friend-VI	Ave.
1. M-Qubed w/ Pepper	5.00	2.00	3.86	5.00	1.99	2.07	2.01	2.01	4.48	2.08	7.00	6.97	**3.70**
2. Always Defect	2.00	2.00	2.00	2.00	2.00	2.08	5.03	2.08	4.51	5.05	7.00	6.38	**3.51**
3. TFT	3.89	2.00	5.00	3.62	4.00	2.06	3.62	2.07	3.75	2.27	5.00	4.42	**3.47**
4. Salt and Pepper	4.89	2.00	2.69	5.00	2.00	1.63	1.63	1.62	2.81	1.64	7.00	6.96	**3.32**
5. FP w/ Pepper	2.02	2.00	3.00	2.00	2.00	2.08	2.11	2.08	4.50	2.08	7.00	6.39	**3.11**
6. GIGA-WoLF w/ Pepper	1.93	1.91	1.99	2.86	1.91	1.99	2.00	1.99	4.44	2.00	6.97	5.55	**2.96**
7. Q-learning	1.98	1.31	3.37	2.71	1.87	1.95	2.15	1.95	4.32	3.12	6.96	3.80	**2.96**
8. Minimax-VI	1.97	1.91	2.00	2.89	1.91	2.00	2.00	1.99	4.45	2.00	6.97	4.78	**2.91**
9. Random	1.52	1.50	3.75	3.49	1.50	1.54	1.71	1.55	3.75	2.40	6.00	5.07	**2.81**
10. WoLF-PHC	1.90	1.21	2.14	2.71	1.89	1.95	1.64	1.94	4.06	1.95	6.94	1.94	**2.52**
11. Always Cooperate	1.00	1.00	5.00	1.00	1.00	0.99	1.00	1.00	3.00	1.02	5.00	3.76	**2.06**
12. Friend-VI	-0.85	-0.94	1.99	-0.89	-0.87	-0.63	-1.20	-0.96	1.01	1.96	2.98	1.93	**0.29**

whereas Pepper and the other learning algorithms in our study learn locally. This prohibits these algorithms from observing global effects not connected in the Markov chain.

Like M-Qubed with Pepper, Salt and Pepper learns to defect against Friend-VI, Always Cooperate, Always Defect, and FP with Pepper. However, as against WoLF-PHC (Figure 3), Salt and Pepper is sometimes exploited by GIGA-WoLF with Pepper and Minimax-VI. Likewise, it does not always defect against Random. Meanwhile, both GIGA-WoLF with Pepper and FP with Pepper learn to defect against all associates, with some slight variations caused by GIGA-WoLF's exploration strategy. These results are consistent with the behavior of these algorithms in the PD matrix game, which demonstrates Pepper's ability to extend algorithms designed for repeated matrix games to RSGs.

We also conducted an evolutionary tournament in the SGPD. In this tournament, an arbitrarily large population of agents, each using one of the 12 algorithms, was evolved over a series of generations according to the algorithms' performance in the SGPD against the agents in the population. Initially, each algorithm was equally represented in the population. In each subsequent generation, the population was altered using the replicator dynamic [23].

Figure 6 shows the proportion of the population using each algorithm over time. After about 10 generations, Tit-for-Tat and M-Qubed with Pepper dominated the population, with Salt and Pepper also holding a small but substantial share. However, once these three algorithms dominated the population, M-Qubed with Pepper quickly took over.

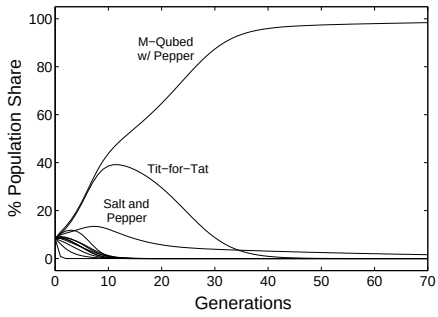


Figure 6: Results of the evolutionary tournament.

6. CONCLUSIONS AND DISCUSSION

In this paper, we presented a new algorithm, called Pepper, which is designed to extend learning algorithms designed for repeated matrix games to algorithms capable of playing effectively in repeated stochastic games (RSGs). To demonstrate the usefulness of Pepper, we extended four matrix learning algorithms from the literature to algorithms for RSGs using Pepper. We then evaluated their performance in a stochastic game prisoner's dilemma (SGPD). Our results show that the behavior of these algorithms in the SGPD is reminiscent of their behavior in the corresponding prisoner's dilemma matrix game.

As in many other MAL algorithms for RSGs, one drawback of Pepper is that all learning is local (within a stage). Thus, it does not account for some effects that are only visible globally. In this paper, this was demonstrated by the fact that none of the learning algorithms we considered were able to learn to consistently cooperate with Tit-for-Tat.

Our results also found that combining the M-Qubed algorithm with Pepper produces an algorithm that learns in the SGPD to cooperate in self play, while learning to defect against associates that are not apt to cooperate. We are not aware of another algorithm in the literature that is able to achieve this without knowing its associate's payoffs as well as the state transitions of the game. As a result, M-Qubed with Pepper outperformed the other algorithms in both a round-robin and evolutionary tournament.

Despite its robust behavior in the SGPD, M-Qubed with Pepper learns very slowly. While its learning rate can be increased to some degree by simply adjusting M-Qubed's learning rate α, perhaps a more effective solution would be to deduce when a conflict between agents is possible [15] or to use different kinds of matrix learning algorithms (MLAs)

in each stage of the game. The selection of the MLA used in each stage could be based on that stage's payoff matrix. Faster matrix learning algorithms could be used in stages that do not appear to require sophisticated reasoning, whereas slower matrix learning algorithms (such as M-Qubed) could be used in stages that appear to require more sophistication. Pepper makes this possible.

7. ACKNOWLEDGMENTS

The author would like to thank Michael A. Goodrich of Brigham Young University for his helpful feedback.

8. REFERENCES

[1] P. Auer, N. Cesa-Bianchi, and P. Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47 (2–3):235–256, 2002.

[2] P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. Gambling in a rigged casino: the adversarial multi-armed bandit problem. In *Proc. of the 36th Symp. on the Foundations of CS*, pages 322–331. IEEE Computer Society Press, 1995.

[3] R. E. Bellman. *Dynamic Programming*. Princeton University Press, NJ, 1957.

[4] B. Bouzy and M. Metivier. Multi-agent learning experiments in repeated matrix games. In *Proc. of the 27^{th} Intl. Conf. on Machine Learning*, 2010.

[5] M. Bowling. Convergence problems of general-sum multiagent reinforcement learning. In *Proc. of the 17^{th} Intl. Conf. on Machine Learning*, pages 89–94, 2000.

[6] M. Bowling. Convergence and no-regret in multiagent learning. In *Advances in Neural Information Processing Systems 17*, pages 209–216, 2005.

[7] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[8] R. I. Brafman and M. Tennenholtz. R-max – a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, March 2003.

[9] E. M. De Cote and M. L. Littman. A polynomial-time Nash equilibrium algorithm for repeated stochastic games. In *Proc. of the 24^{th} Conf. on Uncertainty in Artificial Intelligence*, 2008.

[10] J. W. Crandall and M. A. Goodrich. Learning to compete, compromise, and cooperate in repeated general-sum games. *Machine Learning*, 82(3):281–314, 2011.

[11] D. Fudenberg and D. K. Levine. *The Theory of Learning in Games*. The MIT Press, 1998.

[12] Herbert Gintis. *Game Theory Evolving: A Problem-Centered Introduction to Modeling Strategic Behavior*. Princeton University Press, 2000.

[13] M. A. Goodrich, J. W. Crandall, and J. R. Stimpson. Neglect tolerant teaming: Issues and dilemmas. In *AAAI Spring Symp. on Human Interaction with Autonomous Systems in Complex Environments*, 2003.

[14] A. Greenwald and K. Hall. Correlated Q-learning. In *Proc, of the 20^{th} Intl. Conf. on Machine Learning*, pages 242–249, 2003.

[15] Y. M. De Hauwere, P. Vranx, and A. Nowe. Future sparse interactions: A MARL approach. In *Proc. of the 9^{th} European Workshop on Reinforcement Learning*, 2011.

[16] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. of the 15^{th} Intl. Conf. on Machine Learning*, pages 242–250, 1998.

[17] M. Johanson, N. Bard, M. Lanctot, R. Gibson, and M. Bowling. Efficient Nash equilibrium approximation through Monte Carlo counterfactual regret minimization. In *Proc. of the 11^{th} Intl. Conf. on Autonomous Agents and Multiagent Systems*, 2012.

[18] Rajeeva Karandikar, Dilip Mookherjee, Debraj Ray, and Fernando Vega-Redondo. Evolving aspirations and cooperation. *Journal of Economic Theory*, 80:292–331, 1998.

[19] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of the 11^{th} Intl. Conf. on Machine Learning*, pages 157–163, 1994.

[20] M. L. Littman. Friend-or-foe: Q-learning in general-sum games. In *Proc. of the 18^{th} Intl. Conf. on Machine Learning*, pages 322–328, 2001.

[21] H. Qiao, J. Rozenblit, F. Szidarovszky, and L. Yang. Multi-agent learning model with bargaining. In *The 2006 Winter Simulation Conf.*, pages 934–940, 2006.

[22] J. R. Stimpson and M. A. Goodrich. Learning to cooperate in a social dilemma: A satisficing approach to bargaining. In *Proc. of the 20^{th} Intl. Conf. on Machine Learning*, pages 728–735, 2003.

[23] P. D. Taylor and L. Jonker. Evolutionarily stable strategies and game dynamics. *Mathematical Biosciences*, 40:145–156, 1978.

[24] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.

9. APPENDIX

Table 5: Parameter settings and algorithm specifications.

Pepper	$K = 20$ (for FP) and $K = 5$ (for Salt, M-Qubed, and GIGA-WoLF); $C = 1000$
M-Qubed	$\gamma = 0.95$, $\eta(t) = \frac{0.04 \cdot 1000}{1000 + \max_s \kappa_t(s)}$, $\omega = 1$, $\xi \in [0.1, 0.15]$, $\alpha = 0.1$, $\tau = 10,000$, and $L_i^{\text{tol}} = 500 \cdot \lvert A_i(s)\rvert \cdot \lvert A_i(s)\rvert \cdot \lvert A_{-i}(s)\rvert \, \xi$,
Salt	$\lambda = 0.99$
Fictitious Play	$\gamma_i^{t+1}(a) = \alpha \gamma_i^t(a) + (1-\alpha) I(a, a_{-i})$, where $I(\cdot)$ is the indicator function and $\alpha = \min\left(0.99, \frac{\kappa_i(s)-1}{\kappa_i(s)}\right)$
GIGA-WoLF	$\eta = 0.01$; explores with probability 0.01; uses $R_i(s)$ to estimate the reward gradient
Minimax-VI	Uses Algorithm 1 with $K = 20$; explores with probability 0.01
Friend-VI	Uses Algorithm 1 with $K = 20$; explores with probability 0.01
WoLF-PHC	$\alpha = \frac{1}{100 + \kappa_i(s,\mathbf{a})/10000}$, $\delta = \delta_w = \frac{1}{20000+t}$, $\delta_l = 4\delta_w$, $\forall s, a$, $Q^0(s,a) = \text{rand}\left(0, \frac{r_i^{\max}}{1-\gamma}\right)$, explores with probability 0.01, initial policy is a random mixed strategy
Q-learning	$\alpha = \frac{1}{10 + \kappa_i(s,\mathbf{a})/10000}$, $\gamma = 1$, explores w/ prob. $\max(0.01, 0.2 - \frac{\kappa_i(s)}{100,000})$

Strong Mitigation: Nesting Search for Good Policies Within Search for Good Reward

Jeshua Bratman
Computer Science & Eng.
University of Michigan
jeshua@umich.edu

Satinder Singh
Computer Science & Eng.
University of Michigan
baveja@umich.edu

Richard Lewis
Department of Psychology
University of Michigan
rickl@umich.edu

Jonathan Sorg
Facebook
jdsorg@umich.edu

ABSTRACT

Recent work has defined an optimal reward problem (ORP) in which an agent designer, with an objective reward function that *evaluates* an agent's behavior, has a choice of what reward function to build into a learning or planning agent to *guide* its behavior. Existing results on ORP show *weak mitigation* of limited computational resources, i.e., the existence of reward functions so that agents when guided by them do better than when guided by the objective reward function. These existing results ignore the cost of finding such good reward functions. We define a nested optimal reward and control architecture that achieves *strong mitigation* of limited computational resources. We show empirically that the designer is better off using the new architecture that spends some of its limited resources learning a good reward function instead of using all of its resources to optimize its behavior with respect to the objective reward function.

Categories and Subject Descriptors

H.4 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Experimentation

Keywords

Reinforcement Learning, Planning

1. INTRODUCTION

Reinforcement learning (RL) and decision-theoretic planning approaches for solving sequential decision making problems typically start with an agent's reward function and focus on designing learning/planning architectures for the agent that allow it to efficiently optimize some cumulative measure of the given reward. Recently Singh et al. [1] have argued that for many applications a more accurate and potentially useful way to describe the applied RL setting is as an agent designer who has an *objective* reward function that prescribes *preferences* over agent behavior, so that when building an agent, the designer is free to choose any reward function as *parameters* for the agent. In this view the designer's objective reward function (as preferences) *evaluates* agent behavior while the agent's reward function (as parameters) *guides* the agent's behavior via its planning/learning algorithm. The conventional RL approach confounds these two roles of rewards, evaluation and guidance, by using the objective reward for both. This raises the question of what benefits, if any, might there be to breaking the conventional equality between preferences and parameters? Singh et al. formalize this question via a new *optimal reward problem* (ORP) that defines the optimal reward function as one that—if used to guide agent behavior—ends up maximizing the utility as evaluated by the agent designer's objective reward function. By solving the ORP for classes of agents and environments, Sorg et al. [2] developed a key insight: good reward functions mitigate agent boundedness, i.e., choosing a reward function to guide the agent different from the evaluatory objective reward function can improve agent performance by mitigating inevitable agent limitations (e.g., bounds on architecture or on computational resources).

In this paper we draw a distinction between two ways in which solving the optimal reward problem might be beneficial to the agent designer. *Weak mitigation* is about the existence of good reward functions — ones that improve agent performance compared to using objective reward—however they are found. Past work on solving the ORP has largely focused on demonstrating and understanding weak mitigation. *Strong mitigation* takes into account the cost of finding good reward functions. Intuitively, strong mitigation is demonstrated when it benefits the agent designer to spend a portion of limited computational resources on a search for good reward functions, rather than spending all on a search for policies that optimize objective reward. Achieving strong mitigation is much more practically consequential than weak mitigation[1] and is the focus of this paper.

The contributions of this paper are threefold. 1) We formalize a new distinction between *weak mitigation* and *strong mitigation* of agent limitations. 2) We define a family of abstract architectures we call **N**ested **O**ptimal **R**eward and **C**ontrol or NORC that follows transparently from our def-

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

[1]Achieving weak mitigation may also have practical benefits if the cost of the reward function search is amortized through extended use of the discovered reward function

inition of strong mitigation. 3) We provide concrete examples of NORC architectures and demonstrate empirically that they achieve strong mitigation in some simple domains crafted to illustrate the connection between forms of good reward and the types of agent limitations, as well as on Othello to show scaling and applicability of the NORC architecture.

Other Approaches to Designing Reward Functions. Inverse-RL [3] and preference elicitation [4] consider the problem of determining a reward function, in the former by observing behavior of a human and in the latter from answers to queries asked of a human. Both are different from ORP which assumes that the human's objective reward function is known to begin with. Reward shaping [5], the idea of designing reward functions that accelerate learning, was shown to be a special case of ORP [6] (they also showed the generality of ORP helps). In summary, while these and other approaches to designing rewards exist, prior work focuses on designing rewards from an understanding of the environment and designer's goals without taking into account limitations of the agent (see Definition 1 below).

A key distinctive feature of the ORP is that the agent's limitations are integral to defining rewards. This is a potential strength because it can lead to greater benefit from a good choice of rewards, but it is also a potential weakness because it makes the problem of finding good rewards harder. Previous work on ORP has demonstrated the strength by showing weak mitigation. In this paper we demonstrate that the weakness can be overcome by showing strong mitigation.

2. FROM WEAK TO STRONG MITIGATION

ORP and weak mitigation. The following notation allows us to define the ORP and weak mitigation. At time step t, the agent ($\mathcal{A}$) gets an observation o_t from the environment ($\mathbf{M}$) and takes an action u_t which causes a stochastic transition in the state of the environment. The agent's history at time k is $h_k = o_1 u_1 \ldots o_{k-1} u_{k-1} o_k$. Reward functions are mappings from agent histories to scalar rewards, and are used both to evaluate and guide behavior. Given a history h (of length $|h|$), the designer's or objective utility is $\mathcal{U}^{\mathcal{O}}(h) \stackrel{\text{def}}{=} \frac{1}{|h|}\sum_{t=1}^{|h|} R^{\mathcal{O}}(h_t)$ while the agent's or guidance utility is $\mathcal{U}^{A}(h) \stackrel{\text{def}}{=} \frac{1}{|h|}\sum_{t=1}^{|h|} R(h_t)$, where $R^{\mathcal{O}}$ is the objective reward function and R is the agent's reward function. The designer's objective is to build an agent whose behavior maximizes expected objective utility. The agent's objective is to behave so as to maximize expected guidance utility. Agent $\mathcal{A}$, when guided by reward function R is denoted $\mathcal{A}^R$, and when interacting with the environment produces history $h \sim \langle \mathcal{A}^R, \mathbf{M}\rangle$, where $\langle \mathcal{A}^R, \mathbf{M}\rangle$ is the distribution over histories. The expected objective utility obtained by the designer from the agent is $\mathcal{U}^{\mathcal{O}}_{(\mathcal{A},\mathbf{M})}(R) \stackrel{\text{def}}{=} \mathbb{E}_{h\sim\langle \mathcal{A}^R,\mathbf{M}\rangle}[\mathcal{U}^{\mathcal{O}}(h)]$. The conventional RL agent is $\mathcal{A}^{R^{\mathcal{O}}}$ and achieves expected objective utility $\mathcal{U}^{\mathcal{O}}_{(\mathcal{A},\mathbf{M})}(R^{\mathcal{O}})$. Separating the agent's reward function from the designer's objective reward function leads to the following definition.

DEFINITION 1. ***(ORP [1])***
Given environment $\mathbf{M}$*, agent* $\mathcal{A}$*, and a set of reward functions* $\mathbf{R}$*, choose an optimal reward function*

$$R^* \stackrel{\text{def}}{=} \arg\max_{R\in\mathbf{R}} \mathcal{U}^{\mathcal{O}}_{(\mathcal{A},\mathbf{M})}(R) \stackrel{\text{def}}{=} \arg\max_{R\in\mathbf{R}} \mathbb{E}_{h\sim\langle \mathcal{A}^R,\mathbf{M}\rangle}[\mathcal{U}^{\mathcal{O}}(h)].$$

Note that the set of optimal reward functions is a function of the triplet $\mathcal{A}, \mathbf{M}, R^{\mathcal{O}}$. In other words the limitations of the agent with respect to its environment play a role in defining the optimal reward; this is the formal departure from other (non-ORP) work on designing reward functions.

A number of results are available on ORP including: 1) If $R^{\mathcal{O}} \in \mathbf{R}$, then $\forall(\mathcal{A},\mathbf{M}), \mathcal{U}^{\mathcal{O}}_{(\mathcal{A},\mathbf{M})}(R^*) \geq \mathcal{U}^{\mathcal{O}}_{(\mathcal{A},\mathbf{M})}(R^{\mathcal{O}})$, i.e., using the optimal reward function to guide agent behavior cannot be detrimental to the designer's goals [1], 2) If $R^{\mathcal{O}} \in \mathbf{R}$ and agent $\mathcal{A}$ is unlimited in capability with respect to environment $\mathbf{M}$, then $\mathcal{U}^{\mathcal{O}}_{(\mathcal{A},\mathbf{M})}(R^*) = \mathcal{U}^{\mathcal{O}}_{(\mathcal{A},\mathbf{M})}(R^{\mathcal{O}})$, i.e., solving the ORP only helps limited agents [2], 3) a number of empirical and theoretical investigations that match agent limitations (e.g., depth-limits on planning) with classes of mitigating reward functions (e.g., exploration based rewards) point to possible prescriptions for good reward function choice [2], and 4) a gradient based algorithm PGRD for learning reward functions online for some planning agents [6]. These results demonstrate what we define as weak mitigation.

DEFINITION 2. ***(Weak Mitigation)***
If $\mathcal{U}^{\mathcal{O}}_{(\mathcal{A},\mathbf{M})}(R^*) > \mathcal{U}^{\mathcal{O}}_{(\mathcal{A},\mathbf{M})}(R^{\mathcal{O}})$*, then we say* R^* *weakly mitigates* $\mathcal{A}$*'s limitations in environment* $\mathbf{M}$ *for objective reward function* $R^{\mathcal{O}}$*.*

Crucially, weak mitigation demands only the existence of a good reward function for the $\mathcal{A}, \mathbf{M}, R^{\mathcal{O}}$ of interest. It demands nothing of how easy or hard it is to find good reward functions for it ignores that cost completely. We consider this cost next.

NORC and Strong Mitigation. Solving the ORP involves computation that may be better spent in the agent elsewhere (e.g., in doing deeper planning). Accounting for this cost of finding good rewards is this paper's main contribution, and is effectively a commitment to a new RL agent architecture in which there is both a conventional search for good value-functions/control-policies as well as a search for good reward functions. Because the goodness of a policy is only defined with respect to a reward function the search for good policies has to be nested inside the search for good reward functions. This leads to a two-level nested optimal reward and control (NORC) architecture that is depicted in the right panel of Figure 1. It splits the computational resources available to the designer between a critic-agent (denoted $\mathcal{C}$) that learns/plans good reward-function-policies to guide the actor-agent $\mathcal{A}$, and the conventional actor-agent that learns/plans good control-policies in the actor environment to achieve high reward from the critic-agent.

The departure in NORC is best understood by comparing it to the other two architectures in Figure 1. The conventional RL architecture (left panel) is an artificial agent that simply receives the objective reward from the environment and engages in a search for the best policy with respect to it. For the weak mitigation architecture (middle panel), the designer precomputes a good reward function outside the architecture and builds it into a critic inside the actor-agent (in effect "misleading" the artificial agent concerning the true nature of the objective reward in order to help overcome its limitations). Finally the new NORC agent (right panel), like the conventional agent, simply receives the objective reward from the environment, and then internally partitions the computation between search for a better reward function (the responsibility of the critic-agent) and the search for a good policy (the responsibility of the actor-agent).

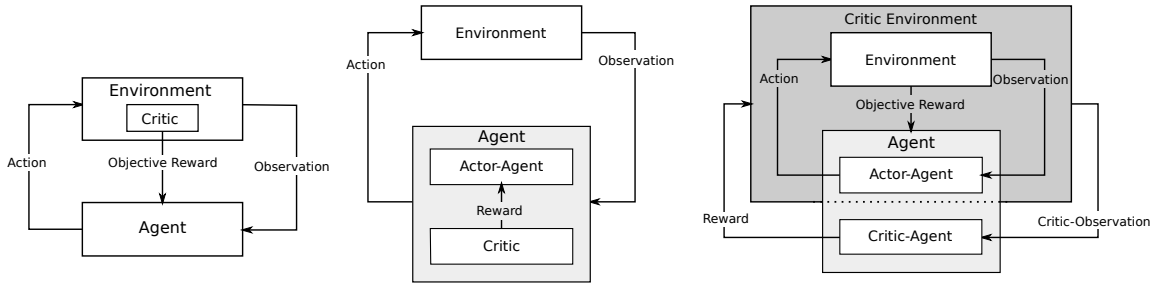


Figure 1: (Left) The conventional RL architecture's agent environment diagram. (Middle) The weak mitigation architecture (from Singh et al. [1]). (Right) The new strong mitigation NORC architecture introduced in this paper. See Section 2 for details.

Intuitively stated, strong mitigation holds when it is better for the resource-bound designer to use a NORC agent rather than a conventional agent, i.e., when learning rewards in addition to learning behavior is a good decomposition of limited resources within an agent. In other words, strong mitigation suggests that reward functions are another important locus of learning or adaptivity within a bounded agent. To formalize this consider a parameterized family of conventional agents, $\mathcal{A}(\Theta)$, where Θ is some set of parameters that impact resource consumption (e.g., one family of agents used in the experiments reported here are finite depth planning agents with depth as resource parameter; larger depth requires more CPU resource per decision). Given a critic-agent $\mathcal{C}$ (we present concrete instances in the next Section), we get a family of NORC agents $\{\mathcal{CA}(\Theta)\}$. Let $Res(\mathcal{A}(\theta \in \Theta))$ be the resource consumption of conventional agent $\mathcal{A}(\theta)$, and $Res(\mathcal{CA}(\theta \in \Theta))$ be the resource consumption of NORC agent $\mathcal{CA}(\theta)$. The expected objective utility obtained by the designer with resource bound τ via the NORC architecture is

$$\mathcal{U}^{\mathcal{O}}_{\mathbf{M}}(\mathcal{CA}(\Theta);\tau) = \underset{\theta\in\Theta \text{ s.t. } Res(\mathcal{CA}(\theta))\leq\tau}{\arg\max} \mathbb{E}_{h\sim\langle\mathcal{CA}(\theta),\mathbf{M}\rangle}[\mathcal{U}^{\mathcal{O}}(h)], \tag{1}$$

and similarly, the expected objective utility obtained via the conventional architecture is

$$\mathcal{U}^{\mathcal{O}}_{\mathbf{M}}(\mathcal{A}(\Theta);\tau) = \underset{\theta\in\Theta \text{ s.t. } Res(\mathcal{A}(\theta))\leq\tau}{\arg\max} \mathbb{E}_{h\sim\langle\mathcal{A}(\theta),\mathbf{M}\rangle}[\mathcal{U}^{\mathcal{O}}(h)]. \tag{2}$$

Definition 3. ***(Strong Mitigation)*** *If $\mathcal{U}^{\mathcal{O}}_{\mathbf{M}}(\mathcal{CA}(\Theta);\tau) > \mathcal{U}^{\mathcal{O}}_{\mathbf{M}}(\mathcal{A}(\Theta);\tau)$, then we say that the critic-agent $\mathcal{C}$ strongly mitigates the limitations of the family of agents $\mathcal{A}(\Theta)$ in environment $\mathbf{M}$ with objective reward function $R^{\mathcal{O}}$ for a resource bound τ.*

We emphasize that weak mitigation (cf. Definition 2) has expected objective utility as a function of reward functions, because that is what the designer chooses in the weak mitigation setting, while strong mitigation (cf. Definition 3) has expected utility as a function of agent architectures (via Equations 1 and 2) because that is what the designer chooses in the strong mitigation setting.

3. CRITIC-AGENTS FOR NORC

Here we turn to providing concrete NORC agents by developing critic-agents; for actor-agents we simply use existing families of popular resource parameterized learning and planning algorithms. The critic-agent's environment is composed of the actor's environment and the actor-agent and thus the state of the critic's environment is (s, s^a) where s is the environment's state and s^a is the actor-agent's state (this is the state of the actor-agent program). The critic-agent's observation consists of both the environment observation o, and the actor-agent's state s^a. The critic-agent's action-space is the set of reward functions $\mathbf{R}$. Recall that the critic-agent is guided by $R^{\mathcal{O}}$ in the NORC architecture while the actor-agent is guided by whatever reward function is provided to it by the critic-agent's action choice

Algorithm 1 General NORC Psuedocode

```
for (t = 1, 2, ...)
  Observe o_t from the environment
    C observes u_{t-1}, R_{t-1}, s^a_{t-1}, o_t, R^O(·_t)
    C chooses reward-function-action R_t ∈ R
      Set A's reward function to R_t
      A observes u_{t-1}, o_t
      A chooses action u_t
  Take action u_t in the environment
```

The NORC architecture involves two interacting exploration/exploitation problems. The first (and inner) is the conventional one faced by the RL actor-agent. The second (and outer) is the new one faced by the critic-agent of whether to stay with the policy for selecting reward functions for the actor-agent that is best based on current knowledge (exploit), or whether to choose reward functions for the actor-agent to improve current knowledge to allow greater objective reward in the future (explore). Unsurprisingly then, the critic-agent is *also* usefully thought of as an RL agent, only with reward functions ($\in \mathbf{R}$) as actions and the joint actor-agent and actor environment as critic environment. This makes it possible to leverage the considerable existing body of algorithms and analysis available for designing RL agents. We consider two different classes of exploration/exploitation problem formulations, the simpler bandits and the more complex MDPs/POMDPs, as the basis for critic-agent algorithms.

3.1 Bandit-based Critic-Agents

First we consider critic-agents that treat their exploration-exploitation problem as a multi-arm bandit problem, i.e., they ignore critic-observations and treat each reward-function-action ($R \in \mathbf{R}$) as an arm and the objective reward as the reward function. Pulling an arm, i.e., selecting a reward-function-action, yields an objective reward through the resulting action choice made by the actor-agent in the environment. In a standard bandit formulation, when an arm

is pulled the reward obtained is an unbiased estimate of the expected utility of that arm, and the choice of arm has no impact on the rewards sampled from other arms in the future. The critic-agent's bandit problem violates these assumptions and thus faces the following challenges: (1) choices of reward-function-action can affect the achievable objective utility for other reward-function-actions by transitioning the critic environment's state (2) samples of immediate objective reward obtained on selecting a reward-function-action are biased estimators for objective-utility for that reward-function-action, and (3) the bias in the objective reward samples can change over time due to dynamics of the actor-agent. We consider two broad classes of bandit critic-agents, corresponding to finite and infinite sets of reward functions.

Finite set of reward functions. When the set of reward functions $|\mathbf{R}|$ is finite (and for practical purposes small) there exist bandit (or experts) algorithms capable of dealing with the exploration-exploitation challenges of the critic-agent via one essential trick, namely that of holding the arm (reward-function-action) fixed for a period of time to alleviate bias and non-stationarity. Of course, how long to hold a reward-function-action fixed is unknown for it depends on the details of the actor-agent and the actor environment. There are several algorithms in the literature that adopt different schemes for incrementally searching for the right length while exploring and exploiting. The algorithm AtEase (*alternating trusted exploration and suspicious exploration*)[7] assumes finite but unknown ϵ-mixing-times, i.e, assumes that if any reward-function-action were held fixed for some unknown, but finite, amount of time the actual average objective reward obtained will be ϵ-close to the expected objective-utility of that reward-function-action choice. A second algorithm, EEE [8], makes no such mixing-time assumptions, and as a result offers weaker guarantees. We focus only on AtEase here.

We do not present the AtEase algorithm in detail here (see [7])), but we sketch a result for an AtEase-critic-based NORC architecture (denoted AtEase-$\mathcal{CA}(\theta)$) that follows directly from Theorem 1 in Talvitie & Singh [7].

Theorem 1. *For environment* $\mathbf{M}$ *and critic-agent reward-function-actions* $\mathbf{R}$*, let* $\mathcal{A}(\theta)$ *be such that NORC architecture* AtEase-$\mathcal{CA}(\theta)$ *satisfies any applicable resource-bounds and for which the finite* ϵ*-mixing-time assumption holds for every* $\mathcal{A}^{R\in\mathbf{R}}(\theta)$*. Then, with high probability* $(1-\delta)$ *for a number of actions* t *polynomial in:* ϵ*-mixing-time of* $\mathcal{A}^{R^*}(\theta)$ *acting in* $\mathbf{M}$*,* $1/\epsilon$*,* $1/\delta$ *and other parameters related to the size of* $\mathbf{M}$*, a history* $h \sim \langle$AtEase-$\mathcal{CA}(\theta), \mathbf{M}\rangle$ *of length* t*, will have objective-utility* $\mathcal{U}^{\mathcal{O}}(h)$ *that is* ϵ*-close to* $\mathcal{U}^{\mathcal{O}}_{\langle\mathcal{A}(\theta),\mathbf{M}\rangle}(R^*)$*.*

In words, if the finite-mixing-time assumption holds, the NORC architecture with an AtEase-critic-agent will, with high probability, and after a number of steps polynomial in the parameters defined above, achieve nearly the same objective-utility as agent $\mathcal{A}^{R^*}(\theta)$ (the weak-mitigation result with apriori-provided optimal reward function R^*). This also means, of course, that the NORC agent will compare favorably with the conventional architecture's agent $\mathcal{A}^{R^{\mathcal{O}}}(\theta)$ (since the conventional agent performs no better than $\mathcal{A}^{R^*}(\theta)$).

Infinite set of reward functions. When the space of reward-function-actions is infinite, the above bandits algorithms do not apply. However, it turns out that PGRD [6] (*policy gradient for reward design*), a recently introduced algorithm for finding good rewards by approximately solving the ORP in differentiably parameterized infinite reward spaces, can be interpreted as a critic-agent algorithm in the NORC architecture. It requires computing the gradient of the objective utility with respect to the reward-function-action parameters. This is only feasible if the actor-agent's procedural mapping from actor-histories to behavior policies (which in turn determine the objective utility) is differentiable. When this condition is satisfied PGRD can treat the actor-agent as a policy parameterized by the reward-function-action parameters and ascend the objective-utility gradient using a policy gradient algorithm. Sorg et al. developed approximate gradient computations for depth-limited planning and UCT[6]; we will use resulting PGRD-critic-agent algorithms for these two families of actor-agents in our experiments.

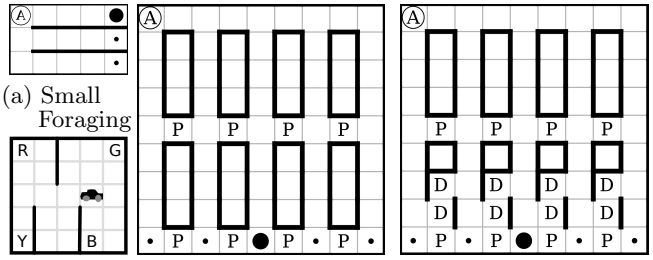
3.2 MDP-based Critic-Agents

The more general case concerns a critic-agent that learns a policy mapping critic-histories to reward-function-actions ($\in \mathbf{R}$). The strongest results in the RL literature for learning policies are in the context of MDPs. When is the critic-agent's exploration-exploitation problem an MDP? The state of the critic environment is given by (s, s^a), and since the state of the actor-agent is always observable, when the actor environment is fully observable, that is $o_t = s_t$, the critic environment is also fully observable (and therefore the critic-agent's problem is an MDP with reward function $R^{\mathcal{O}}$). However, if the actor-agent's dynamics are non-stationary, then the critic's problem will be a non-stationary MDP. For depth-limited planning and UCT with fixed and given models, the actor-agent's dynamics are stationary. In such settings, any algorithm for solving MDPs can be used as a critic-agent algorithm. In particular, simple RL algorithms such as ϵ-greedy Q-learning become applicable (and for finite-state critic environment and a finite set of reward-function-actions, are guaranteed to converge to the optimal reward-function-action policy). In cases where either the actor environment is not fully observable or the actor-agent's dynamics are non-stationary, more sophisticated RL algorithms can be used, but we do not explore those here.

4. EXPERIMENTS

The following set of experiments together are intended to show the robustness of the strong mitigation phenomenon across different families of actor-agents (depth-limited planning, the recent TDPlanning algorithm [9], and state-of-the-art UCT), across different kinds of actor-agent limitations (limited-depth, incorrect modeling assumptions, limited sampling and search control) as well as across different spaces of reward functions (those based on domain independent features such as inverse-recency and inverse-frequency, as well as those based on domain-dependent sub-goal features).

Common Structure in Experiments. For each experiment we will describe 1) the environment and objective reward function, 2) the family of resource-parameterized planning algorithms used both as conventional agents and as actor-agents for NORC, 3) the critic-agent algorithms used, and 4) the space of reward functions that determine the actions available to the critic-agents. In all experiments, the resource bounds we consider are limits on the *CPU-time per decision* and we compare the conventional agent with mul-



(a) Small Foraging (d) Taxi (b) Foraging (c) Darkroom Foraging

Figure 2: Illustration of environments. Thick black lines are impassable walls. In the foraging domains (a,b,d), small circles represent inexhaustible food, the large circle is one possible location for the consumable food, 'P' are pits, and 'D' are dark rooms.

tiple NORC agents for various limits on time per decision. Intuitively, in each experiment we expect a threshold on time per decision below which the NORC agent will outperform the conventional agent, demonstrating strong mitigation.

4.1 Learning in Bandit-Based Critic-Agents

Our first set of results are for critic-agents using the bandit algorithms described above (for finite sets of reward functions) and PGRD (for continuous reward functions).

4.1.1 Bounded planning depth

Here we contrast weak and strong mitigation for a family of actor-agents directly parameterized by planning depth (which indirectly parameterizes time per decision).

Environment and objective reward: An agent navigates a 5×3 grid (shown in figure 2a) choosing among actions: *North, South, East, West,* and *Eat*. Movements fail with probability 0.1 resulting in a random movement in any other direction. At the end of each corridor is an inexhaustible food source which when eaten gives objective reward +0.01. A second, consumable, food is at the end of one of the three corridors which when eaten provides +1 and is then replaced by another at the end of a different corridor.

Family of actor-agents: Agents employing full width, limited depth planning, with parameters $\Theta = \{d\}$ for planning depths $d \in \{3, \ldots, 11\}$ (which determine CPU-time per decision).

Critic-agents: Algorithms described in section 3: ATEASE, and PGRD[2]. We also included other bandit algorithms for comparison: EEE [8], EXP3 [10], and ϵ-greedy.

Reward functions: The reward functions were linear combinations of three domain-independent features: (1) inverse-recency (used e.g., by [2]) given by $\boldsymbol{\phi}_{\text{inv-rec}}(h) = 1 - 1/c(o_{|h|})$ where $c(o_{|h|})$ is the number of steps since observing the current observation $o_{|h|}$ (recall this domain is fully observable so this is the same as the number of steps since visiting the current state $s_{|h|}$). A reward function with a positive coefficient on this feature encourages the actor-agent to visit less-recently-visited states, resulting in persistent exploration. (2) inverse-frequency given by $\boldsymbol{\phi}_{\text{inv-freq}}(h) = 1/n(o_{|h|-1}, u_{|h|-1})$ where $n(o_{|h|}, u_{|h|})$ is the total number of times the agent has taken action $u_{|h|}$ after observing $o_{|h|}$. A positive coefficient on this feature encourages the actor-agent to visit less-frequently-visited transitions. (3) distance-to-goal heuristics, specifically we used Manhattan distance to the consumable food ($\boldsymbol{\phi}_{\text{dist}}(h)$). The reward function space is composed of linear functions:

$$R(h) = R^{\mathcal{O}}(h) + \theta_1 \boldsymbol{\phi}_{\text{inv-rec}}(h) + \theta_2 \boldsymbol{\phi}_{\text{inv-freq}}(h) + \theta_3 \boldsymbol{\phi}_{\text{dist}}(h)$$

The bandit algorithms used a coarse discretization of the parameter space with $(\theta_1, \theta_2, \theta_3) \in \{-1, -0.1, 0, 0.1, 1\}^3$ yielding a total of $|\mathbf{R}| = 5^3$ reward functions, while PGRD optimized over the continuous space $(\theta_1, \theta_2, \theta_3) \in \mathbb{R}^3$.

[2]ATEASE used $\epsilon = \infty$ (so it is not suspicious when exploiting) and $l = 90$, PGRD used discount $\gamma = 0.99$, temperature 100, and learning rate $\alpha = 5^{-5}$ (see references for details).

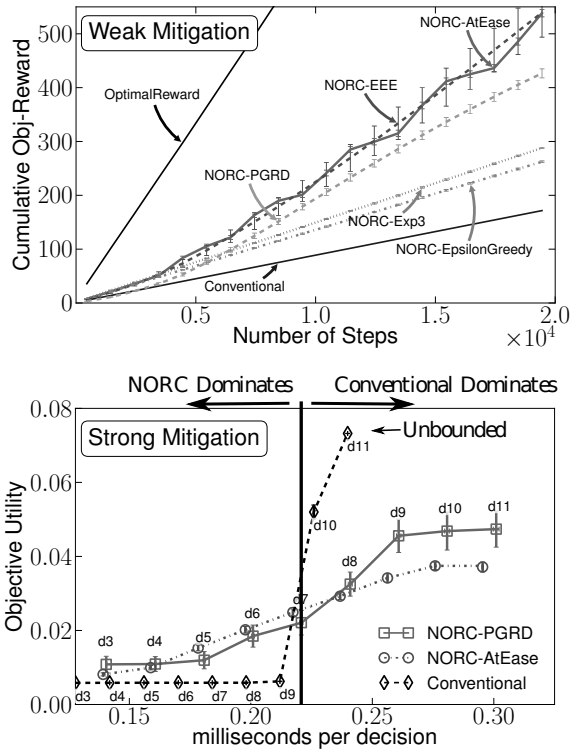


Figure 3: Experiments on small foraging domain; (top) compares single conventional agent — a depth 5 planner — with corresponding NORC agents. All NORC agents outperform the conventional agent showing weak mitigation, but not strong mitigation. (bottom) shows both objective utility and computational resources required for a family of actor-agents and corresponding NORC agents. For a resource bound less than that indicated by the vertical line, NORC agents outperform conventional agents, thus showing strong mitigation.

Results Figure 3 (top) shows performance as a function of number of steps for an actor-agent with planning depth 5. Unsurprisingly Agent $\mathcal{A}^{R^*}$ (denoted OptimalReward) did the best since it was given the optimal reward function for guidance from the start. In particular it did far better than the conventional agent that used the objective reward; this is a weak mitigation result because it ignores bounds on CPU-time. Interestingly, all NORC agents obtained greater cumulative objective reward than the conventional agent. This sets up the possibility of strong mitigation which we explore in Figure 3 (bottom) that shows objective-utility (over 20,000 steps) plotted against CPU-time per decision (τ) for three agents. For τ less than about $0.22ms$ the agent designer is better off choosing an agent with smaller plan-

ning depth using the NORC architecture than choosing a conventional agent with larger planning depth. This is the key strong mitigation result. As the resource bound gets looser ($\tau > 0.22ms$), choosing the conventional agent becomes better (in fact, at depth 11, the conventional agent acts optimally).

4.1.2 Bounded sample-based planning

This experiment shows strong mitigation for a second family of actor-agents using the popular and efficient Monte Carlo tree-search algorithm UCT [11].

Environment and objective reward Similar to Experiment 1, but larger, and with the addition of pits that transport the agent to a random location in the top row (see Figure 2b), eating the inexhaustible food yields objective reward +0.001.

Family of actor-agents: UCT with parameters $\Theta = \{(d, t)\}$ for planning depths $d \in \{5, 15, 35\}$ and trajectory counts $t \in \{50, 100, 500, 1000\}$; together these parameters determine CPU time per decision over 20,000 steps. UCT builds a search tree by simulating (from the current state) t trajectories of depth d where decisions in the trajectory generation are treated as a bandit problem solved with the UCB1 algorithm. We performed two experiments, one in which the actor-agent was given a perfect environment model and a second in which the actor-agent learned a model using the empirical probabilities observed in the data. To provide exploration, the model-learning actor-agents had a 0.1 probability of taking a random action.

Critic-agents: AtEase, and PGRD with the same parameters as in the previous experiment.

Reward functions: The form of the reward function was similar to the previous experiment, but the Manhattan distance heuristic feature was replaced by a model error feature $\boldsymbol{\phi}_{\text{model-error}}$ measuring inaccuracy of the agent's model for each transition (as described by Sorg et al. [2]). Reward functions with positive coefficients on the model-error feature encourage an agent to explore less-well-modeled transitions. AtEase used the same discretization of the parameter space as before.

Results: Results are in Figure 4a and 4d. When the model was given, for CPU-time per decision τ less than about $8ms$, choosing a NORC agent is better than a conventional agent. Again, showing strong mitigation, in this case for UCT-based actor-agents. When the model was learned, for the range of t, d parameters explored, the NORC agents dominate conventional agents in part because the reward functions provided by the critic-agent improved model-learning speed through rewarding persistent exploration (this is evident in that for the largest CPU-time per decision, the NORC agents achieved objective-utility much closer to their given-model counterparts than did the conventional-agent).

4.1.3 Incorrect model representation

In this experiment we show strong mitigation for UCT-based actor-agents with an incorrect modeling assumption in the model (both learned and given).

Environment and objective reward Like previous experiment except the environment is partially observable: between each corridor is a 2-square dark room through which the agent can move as normal, but cannot distinguish its location. See Figure 2c.

Actor-agents, critic-agents, rewards Like previous experiment; both the given and learned models made the first-order Markov assumption which is incorrect due to the dark-room. Also, notice negative coefficients on the model error feature encourage agent to avoid the dark room transitions.

Results: Results are in Figure 4b and 4e. When the model was given, the PGRD-NORC agents performed significantly better than the conventional agent for $\tau < 2ms$ CPU-time per decision. Evidence of strong mitigation again, in this case with the additional limitation of incorrect modeling assumptions in the face of partial-observability. For the second experiment when the actor-agent learned a first-order Markov model, the NORC agents dominated conventional agents for all parameters tested (as in the learned model instance of the previous Experiment for similar reasons).

4.2 Learning in MDP-based Critic-Agents

Here we depart from previous experiments with bandit-based critic-agents to explore MDP-based critic-agents that learn reward-function-action policies conditioned on abstractions of critic-histories.

Environment and objective reward: The taxi domain (seen in Figure 2d) was introduced by Dietterich [12]. The agent controls a taxi tasked with delivering a passenger from one of four pickup locations (labeled R,Y,G,B; random on each episode) to one of four (also random) destinations. There are six actions: *North, South, East, West, Pick Up Passenger,* and *Drop Off Passenger*. Objective reward gives +1 if passenger is dropped off at the correct destination (in which case the episode ends), -0.5 if dropped off incorrectly, and -0.05 on other transitions. State is fully observable and factored into three features $s = (\boldsymbol{\phi}_{taxi}, \boldsymbol{\phi}_{pssngr}, \boldsymbol{\phi}_{dest})$.

Family of actor-agents: Planning agents using TDPlanning [9] which is similar to UCT, but rather than building a search tree, it learns a value function[3] through t simulated trajectories of maximum length d. Depths and transition count parameters Θ were identical to the UCT experiments above. We also performed the same experiment using UCT actor-agents with similar results (not shown).

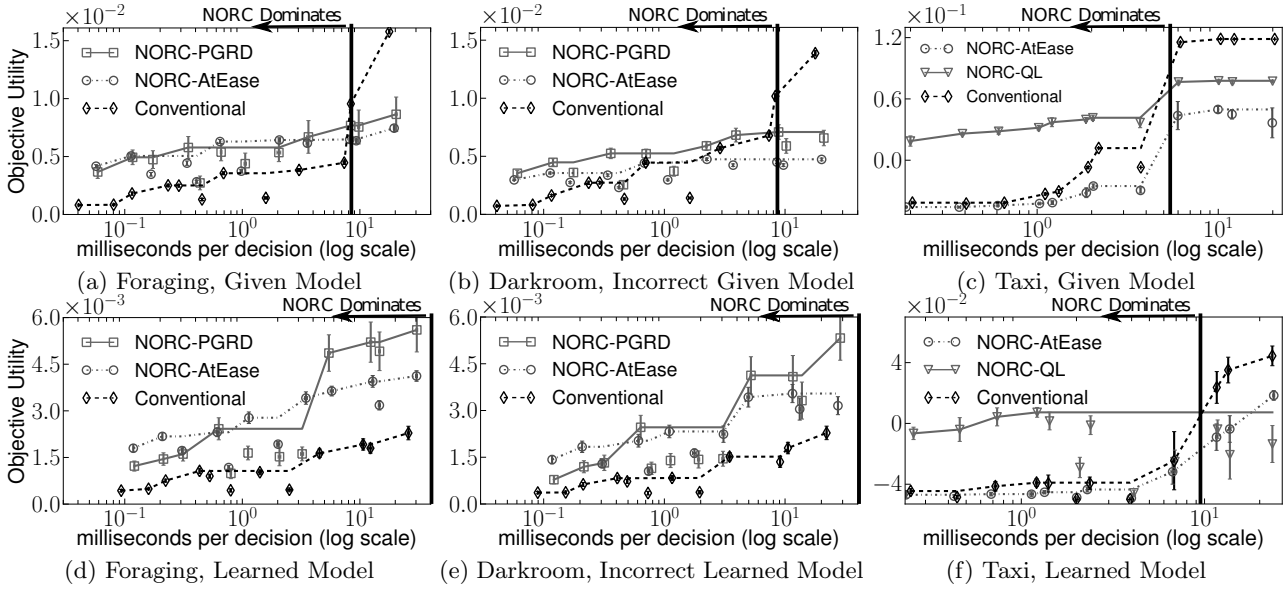
Critic-agents: ϵ-greedy Q-Learning[4] (actor-agent had no persistent state so critic environment state was actor environment state), and AtEase with parameters before.

Reward functions: Each was a single reward feature multiplied by a scalar for the inverse-recency feature $\boldsymbol{\phi}_{\text{inv-rec}}$ and state features $\boldsymbol{\phi}_{taxi}$ and $\boldsymbol{\phi}_{pssngr}$. Let δ be the Kronecker delta function, then we have one reward functions for each taxi location: $\delta(\boldsymbol{\phi}_{taxi}, l) \forall l \in \{1, \cdots, 25\}$, each passenger location $\delta(\boldsymbol{\phi}_{pssngr}, p) \forall p \in \{1, 2, 3, 4, 5\}$, for a discrete set of weights on the inverse-recency $\theta \boldsymbol{\phi}_{\text{inv-rec}} \forall \theta \in \{-1, -.1, 0, .1, 1\}$, and the objective reward function $R^{\mathcal{O}}$ totaling 36 reward functions (actions for the critic-agent).

Results: Results are in Figure 4c and 4f. The critic-agent using Q-Learning successfully learned a policy over reward functions allowing the NORC-QL agent to perform much better than the conventional agents for CPU-time limitations τ less than about $4ms$. However, NORC-AtEase where the critic-agent learned an unconditional reward policy, did not perform well because no single reward function in the set is particularly useful unless chosen as a function of state. When the actor-agent learns a model, the NORC-QL

[3] In our TDPlanning algorithm, the value function was learned with ϵ-greedy Q-Learning, $\epsilon = 0.1$, learning rate $\alpha = 0.4$

[4] $\epsilon = 0.1$ and learning rate of $\alpha = 0.9$



(a) Foraging, Given Model (b) Darkroom, Incorrect Given Model (c) Taxi, Given Model

(d) Foraging, Learned Model (e) Darkroom, Incorrect Learned Model (f) Taxi, Learned Model

In the graphs above, the points for each agent read from left to right correspond to actor-agent parameters:
(depth, trajectories): (5,50),(5,100),(15,50),(15,100), (35,50),(5,500),(35,100),(5,1000),(15,500),(15,1000),(35,500),(35,1000)

Figure 4: (left column) strong mitigation with bounded sample-based planning actor-agents (UCT) and critic-agents learned unconditional selection of reward functions; (center column) same as experiment in left column, but actor-agents had incorrect modeling assumption; (rightmost) strong mitigation in Taxi where the NORC-QL critic learned a policy over reward functions (NORC-AtEase did not, and as a result performed poorly in this experiment). The line drawn for each agent shows the highest score achieved up to a given resource level (this is a visual aid approximating the best performance per cpu time across parameters).

agent does not perform well, this may be because the non-stationarity of the actor-agent misleads the QL critic-agent.

These results show strong mitigation when the critic-agent learns a policy over reward-function-actions. In this experiment, the critic-agent essentially chose subgoals that could be achieved by an actor-agent using far less computational resources than necessary to act well given only the objective reward.

5. STRONG MITIGATION IN OTHELLO

Our final results show the practical import of NORC and strong mitigation on the board game of Othello, a large domain with a long history in AI.

Environment and objective reward: We experiment using both a 6×6 board and the standard 8×8 board. The 8×8 game has roughly 10^{28} states and a large branching factor determined by the number of legal moves at each step. For our experiments, the agent played against a fixed opponent agent planning with conventional UCT (depth 20 and trajectory count 100). The objective reward was $+1$ for a win, 0.5 for a draw, and 0 for a loss. The agents were evaluated and learned while playing against an opponent rather than learning with self play (a common paradigm in computer game playing).

Family of actor-agents: UCT with parameters given by the 20 combinations of depths $\{5, 10, 15, 20, 25\}$ and trajectory limits $\{50, 100, 250, 500\}$. We did not provide the agent with an opponent model, instead UCT generated a minimax planning tree by choosing actions for both black and white players as described by Gelly and Silver [13] for their master level computer GO player.

Critic-agent: PGRD-UCT with learning rate 10^{-7} and temperature parameter 100.

Reward functions: We used the weighted piece counter (wpc) [14] feature representation where $\boldsymbol{\phi}_{wpc}(o)$ consists of 64 features, one for each location on the board, with value $+1$ if that location contains a black piece 0 if empty and -1 if white. Reward functions were linear combinations of these features evaluated on the state reached after taking an action: $R(h) = R(o, a, o') = \theta^T \boldsymbol{\phi}_{wpc}(o')$.

Results: Each NORC and conventional agent was evaluated over 12 trials of 30,000 games; for half of these trials our agent plays white and for the other half black. Results are shown in Figure 5. It is immediately clear that the NORC agents required much more computation than the corresponding conventional agents (with same (d, t) parameters). For example, with depth 25 and trajectory count 500 the NORC agent required nearly twice the time per decision as the corresponding conventional agent. Nevertheless, we see strong mitigation: the reward functions learned by the critic-agent allow the UCT actor-agent to achieve higher quality planning with less computation, so given equivalent CPU time, a designer is better off choosing a NORC agent. Learned reward functions can improve UCT in a variety of ways such as providing better terminal evaluation and by improving search control via encouraging exploration of more

[5]For each agent, reading points from left to right on the graph, the parameter pairs are $(depth, trajectories)$:(5,50), (10,50), (5,100), (15,50), (20,50), (25,50), (10,100), (15,100), (20,100), (5,250), (25,100), (10,250), (15,250), (5,500),(20,250), (25,250), (10,500), (15,500), (20,500), (25,500)

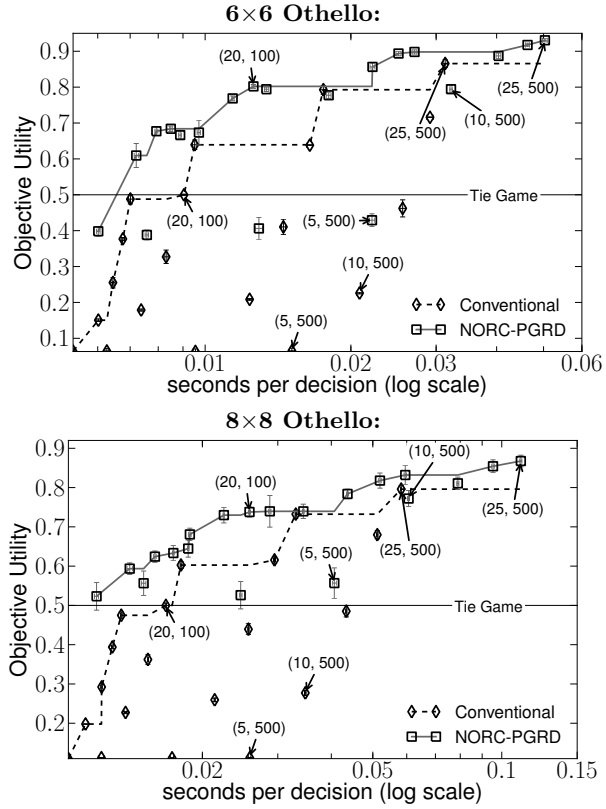


Figure 5: Othello results. NORC agents required significantly more computation than the corresponding conventional agents, but allowed the bounded UCT planning actor-agents to choose better actions even with less computational time (fewer and shallower sample trajectories). Several parameter pairs $(depth, trajectories)$ are labeled[5].

fruitful trajectories in planning (as discussed by Sorg et. al. [6]). Furthermore, in these experiments the NORC agents are more robust across the space of parameters.

6. CONCLUSION

The previously defined optimal reward problem (ORP) takes the agent's limitations expressed in its environment into account when designing rewards; this was in contrast to other approaches for designing rewards. Building on ORP this paper distinguishes between weak mitigation, the subject of previous ORP work, and the more practical strong mitigation developed here. We showed how strong mitigation implies a nested optimal reward and control (NORC) architecture and developed concrete NORC instantiations that allowed us to demonstrate over a variety of actor-agent planning algorithms that it can be better to nest the search for good policies inside a search for good reward functions. Our strong mitigation results on Othello demonstrate the scalability of the NORC architecture. Finally, our results with NORC suggest it might be beneficial to treat reward functions as a locus of learning and adaptivity within an autonomous agent – just as it might be beneficial to learn value functions or policy functions.

Acknowledgments.

This work was supported by NSF grants IIS-0905146 and IIS-1148668. Any opinions, findings, conclusions, or recommendations expressed here are those of the authors and do not necessarily reflect the views of the sponsors.

7. REFERENCES

[1] Satinder Singh, Richard L. Lewis, Andrew G. Barto, and Jonathan Sorg. Intrinsically motivated reinforcement learning: An evolutionary perspective. *IEEE Transactions on Autonomous Mental Development*, 2010.

[2] Jonathan Sorg, Satinder Singh, and Richard Lewis. Internal rewards mitigate agent boundedness. In *Proceedings of the International Conference on Machine Learning*, 2010.

[3] Andrew Y. Ng and Stuart J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the International Conference on Machine Learning*, 2000.

[4] Urszula Chajewska, Daphne Koller, and Ronald Parr. Making rational decisions using adaptive utility elicitation. In *In Proceedings of the National Conference on Artificial Intelligence*, 2000.

[5] Andrew Y. Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the International Conference on Machine Learning*, 1999.

[6] Jonathan Sorg, Satinder Singh, and Richard Lewis. Optimal rewards versus leaf-evaluation heuristics in planning agents. In *Proceedings of the Twenty-Fifth Conference on Artificial Intelligence*. 2011.

[7] Erik Talvitie and Satinder Singh. An experts algorithm for transfer learning. In *Proceeding of the International Joint Conference on Artificial Intelligence*, 2007.

[8] Daniela Pucci De Farias and Nimrod Megiddo. Combining expert advice in reactive environments. *Journal of the ACM*, 2006.

[9] David Silver, Richard S. Sutton, and Martin Müller. Sample-based learning and search with permanent and transient memories. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[10] Peter Auer, Yoav Freund, and Robert E. Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Proceedings of the Symposium on Foundations of Computer Science*, 1995.

[11] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *Proceedings of the European Conference on Machine Learning*, 2006.

[12] Thomas G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 2000.

[13] Sylvain Gelly and David Silver. Achieving master level play in 9 x 9 computer go. In *Proceedings of the Conference on Artificial Intelligence*, 2008.

[14] P. Hingston and M. Masek. Experiments with Monte Carlo Othello. In *Proceedings of the IEEE Congress on Evolutionary Computation*, 2007.

Session 2C
Learning II

Decentralized Bayesian Reinforcement Learning for Online Agent Collaboration

W. T. L. Teacy[1], G. Chalkiadakis[2]
A. Farinelli[3]
[1]University of Southampton, UK
{wtlt,acr,nrj}@ecs.soton.ac.uk
[2]Technical University of Crete, Greece
gehalk@intelligence.tuc.gr

A. Rogers[1], N. R. Jennings[1]
S. McClean[4], G. Parr[4]
[3]University of Verona, Italy
alessandro.farinelli@univr.it
[4]University of Ulster, UK
{si.mcclean,gp.parr}@ulster.ac.uk

ABSTRACT

Solving complex but structured problems in a decentralized manner via multiagent collaboration has received much attention in recent years. This is natural, as on one hand, multiagent systems usually possess a structure that determines the allowable interactions among the agents; and on the other hand, the single most pressing need in a cooperative multiagent system is to coordinate the local policies of autonomous agents with restricted capabilities to serve a system-wide goal. The presence of uncertainty makes this even more challenging, as the agents face the additional need to learn the unknown environment parameters while forming (and following) local policies in an online fashion. In this paper, we provide the first Bayesian reinforcement learning (BRL) approach for distributed coordination and learning in a cooperative multiagent system by devising two solutions to this type of problem. More specifically, we show how the Value of Perfect Information (VPI) can be used to perform efficient decentralised exploration in both model-based and model-free BRL, and in the latter case, provide a closed form solution for VPI, correcting a decade old result by Dearden, Friedman and Russell. To evaluate these solutions, we present experimental results comparing their relative merits, and demonstrate empirically that both solutions outperform an existing multiagent learning method, representative of the state-of-the-art.

Categories and Subject Descriptors

I.2.6 [**Learning**]; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Algorithms

Keywords

multiagent learning, Bayesian techniques, uncertainty

1. INTRODUCTION

In cooperative multiagent systems, the grand challenge is to ensure that a common, system-wide goal is achieved by coordinating the actions of individual agents. Often, however, this is difficult because each agent (1) only has a limited world view, and (2) has no direct control over the actions of its peers. Agents are therefore restricted to forming local policies subject to local information. In such cases, the goal is to coordinate the agents' local polices to form a joint optimal one—a problem that is, in general, computationally infeasible [2]. However, multiagent systems usually possess some form of structure, which can often be exploited to perform efficient coordination. For example, in Distributed Constraint Optimisation Problems (DCOPs), an agent's actions are only dependent on a subset of its peers—a fact that can be used to construct efficient coordination algorithms for the agents as a whole [9].

Indeed, solving such complex but structured problems is challenging, particularly in the context of reinforcement learning (RL) [18], in which agents must *explore* their environment to learn how best to act. However, existing collaborative reinforcement learning techniques [11, 13] are "point-based", i.e. they do not optimize decisions w.r.t. all possible world models. For this reason, they provide a suboptimal solution to the *exploration-exploitation* problem [18], in which agents must decide when to explore actions of uncertain value, which may yet prove to be optimal.

This is particularly important for problems involving real hardware, e.g. Unmanned Aerial Vehicles (UAVs), for two reasons: (1) repeated trials may be expensive and time consuming, and so control software must learn effectively from few interactions with the environment; and (2) exploratory actions may result in damage or injury, and so one must account for respective risks and value.

Here, we address this by making the following three key contributions to the state-of-the-art: (1) in order to provide a near optimal solution to the exploration-exploitation trade-off, we present the *first* model-free and model-based algorithms for decentralised Bayesian Reinforcement Learning (BRL) in a cooperative multiagent system; (2) by empirical analysis, we show that both algorithms *outperform* an existing state-of-the-art decentralised learning method, while at the same time provide different complementary trade-offs between computational complexity, and the amount of exploration required to learn an effective coordination strategy; (3) as part of our model-free method, we provide *a closed-form solution for the Value of Perfect Information (VPI)*, which we use to perform efficient exploration. The latter corrects a key result by Dearden, Friedman and Russell, central to their original contribution on Bayesian Q-learning [7].

In the rest of the paper, Sections 2 and 3 outline related work; Section 4 presents our closed-form solution for VPI; Section 5 describes how this can be used for decentralised BRL; Section 6 evaluates these algorithms empirically; and Section 7 concludes.

2. DECENTRALIZED COORDINATION

A decentralized coordination problem is one in which a set of agents must together choose how to act so that their joint utility is maximized. Here, we outline a solution to this class of problems, based on the use of the max-sum algorithm. Specifically, given a factored utility function $F(\mathbf{a}) = \sum_i F_i(\mathbf{a}_i)$, the goal of decentralized co-

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

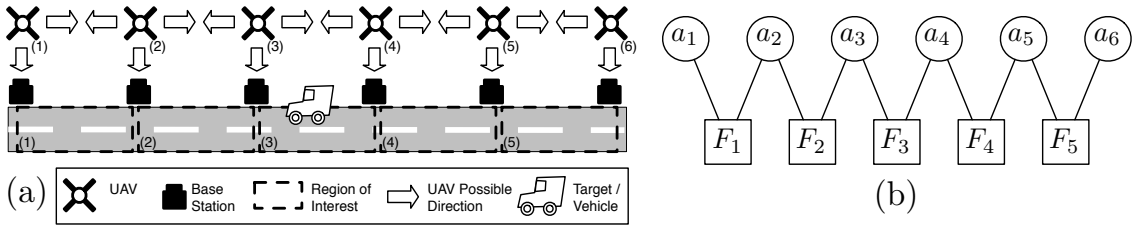


Fig. 1 (a) Six UAVs patrol a road for vehicles. Each has a base station where it can land during idle periods, and is responsible for patrolling its adjacent regions, east and west along the road. When a vehicle is detected in a region (e.g. by ground based motion sensors), the pair of UAVs bordering the region are alerted, and must both patrol the region simultaneously to observe the vehicle. **(b)** The corresponding factor graph with factors represented by squares, and actions by circles.

ordination is to find the joint action vector, $\boldsymbol{a}$, that maximizes the global utility function:

$$\arg\max_{\mathbf{a}} \sum_i F_i(\mathbf{a}_i) \tag{1}$$

Here, each $F_i(\mathbf{a}_i)$ represents a local utility function (*a factor*), and $\mathbf{a}_i \subseteq \mathbf{a}$ are *local action vectors*.[1] This is efficiently solved by the *max-sum* algorithm; a technique of the Generalized Distributive Law (GDL) [1], widely used for computing factored functions using local message passing [19].

In particular, the factored optimization problem described in Eq. 1 can be viewed as a DCOP and represented by a bipartite factor graph [14]. For example, the scenario illustrated in Fig. 1(a) can be represented by the bipartite factor graph in Fig. 1(b). Specifically, this represents the factored *reward* function $F = \sum_{i=1}^{5} F_i(a_i, a_{i+1})$, where each factor node, F_i, represents the local reward for observing a given region of road, the action nodes represent the decision variables for each UAV, and edges connect factors to the actions on which they depend. The max-sum algorithm operates on the factor graph by iterative message passing between neighbouring variable and factor nodes. When the graph is cycle-free, the messages are guaranteed to converge and the global optimal solution is computed. While no such guarantees exist for cyclic graphs, extensive empirical evidence demonstrates that good solutions can still be reached [9].

Although the max-sum on its own can be applied in a variety of settings, the coordination problem as defined above only deals with cases in which agents must choose a single joint action to receive a single immediate reward. However, in sequential decision making, agents must also consider their action's effect on the future world state, which in turn influences their future rewards. In detail, consider an agent's decision-making problem in a stochastic environment modeled as a Markov Decision Process (MDP) $\langle \mathcal{S}, \mathcal{A}, \Pr, R \rangle$, with finite state and action sets $\mathcal{S}, \mathcal{A}$, transition dynamics $\Pr$, and reward function $R : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, where $\Pr(s'|s,a)$ is the probability of reaching state s' after taking action a at s. Similarly, $R(s,a)$ denotes the expected reward which is obtained when action a is performed at state s. The agent then needs to construct an optimal policy, $\pi : \mathcal{S} \mapsto \mathcal{A}$, derived by solving a system of Bellman equations [18]:

$$Q(s,a) = R(s,a) + \gamma \sum_{s' \in \mathcal{S}} Pr(s'|s,a) Q(s', \pi(s)) \tag{2}$$

Here, γ is a discount factor that places more weight on immediate rewards; the *Q-value* function, $Q : \mathcal{S} \times \mathcal{A} \mapsto \mathbb{R}$, is equal to the expected total sum of future discounted rewards given the current state and action; and from this, the optimal policy is $\pi(s) = \arg\max_a Q(s,a)$. This standard formulation was originally extended to multiagent MDPs by defining $\mathcal{A}$ as the cartesian product of individual action spaces associated with each agent [3]. Thus, $\boldsymbol{a} \in \mathcal{A}$ is a joint action vector comprising individual agents' actions. Note that actions are chosen in the context of a state, which is represented here as a vector $\boldsymbol{s} \in \mathcal{S}$. That is, $\boldsymbol{s}$ comprises global state variables shared by all agents, as in a direct extension of the classic factored state representation [4], but may also contain state variables specific to individual agents, such as a UAV's battery life.

In principle, such multiagent MDPs may be solved like any other MDP, except that by taking the cartesian product of individual action spaces, the computational complexity is exponential in the number of agents. Fortunately, in many coordination problems (e.g. Fig. 1), the actions of each agent are only strongly dependent on a subset of their peers. In such cases, good approximations to the optimal policy can often be found by representing the problem as a *factored* MDP [10]. Specifically, these can be defined by assuming a factored structure for the Q-value function, $Q(\boldsymbol{s}, \boldsymbol{a}) = \sum_i Q_i(\boldsymbol{s}_i, \boldsymbol{a}_i)$, where each factor, Q_i, depends only on a subset of states, $\boldsymbol{s}_i \subseteq \boldsymbol{s}$, and actions $\boldsymbol{a}_i \subseteq \boldsymbol{a}$. Using this assumption, algorithms for solving factored MDPs can make efficiency savings similar to those made by max-sum. For example, [10] presents a dynamic programming solver for factored MDPs, which can converge in approximately logarithmic time, despite the state-action space growing exponentially.

While, in general, assuming a factorisation of the Q-value function might result in suboptimal solutions, a trade-off between complexity and optimality can be found by decomposing the problem in different ways. For example, at one extreme, associating a factor with each agent favours computational efficiency by achieving a large degree of factorisation. In contrast, we guarantee an optimal solution when no factorisation is performed, but the complexity of finding this solution is now exponential in the total number of state and action variables.

Here, however, we adopt a more graded approach, by decomposing coordination problems into *regional* subproblems. Specifically, we associate each factor with a *region*, which can represent any part of the overall decision problem that depends only on a subset of agents. For example, in Fig. 1, each region represents part of the road, in which observations are only made by neighbouring UAVs. However, we could equally decrease the amount of decomposition by aggregating adjacent regions. In principle, this would enable better polices at the expense of more computation, because each factor would now account for dependencies between the actions of a larger number agents [10].

3. BAYESIAN RL

In standard Reinforcement Learning (RL), a single agent is faced with a decision problem, typically modelled as an MDP with *unknown* reward and transition dynamics. Due to this additional uncertainty, the agent cannot simply solve the MDP to find the best policy, but must instead *learn* it by *exploring* different states and ac-

[1] We abuse notation here slightly by applying set operators to vectors. The intended interpretation is that a vector's elements form a set, related to another vector's elements.

tions. To achieve this, classical RL techniques require some form of heuristic to encourage learning, by exploring actions with unknown outcomes. Generally, however, these heuristics are based on intuition alone, and so lack theoretical foundations based on any notion of optimality.

In contrast, Bayesian RL (BRL) methods [5] formulate the problem as a *belief-state MDP*, in which an agent's beliefs are explicitly modelled as part of the state. In this way, an agent can *infer* the exploratory value of an action, by reasoning about how the information obtained by performing an action may enable better future decisions. The solution to the belief-state MDP provides the optimal solution to the action selection problem, taking full account of the informative value of exploratory actions. Therefore, BRL *does not* require the use of an explicit exploration heuristic; instead, agents need only act greedily w.r.t. the Bayesian Q-values to achieve optimal learning. No other method outperforms the Bayesian one in expectation, when using the same prior information [15].

Unfortunately, solving the belief-state MDP is, in general, computationally infeasible. Nevertheless, the Bayesian approach does provide a theoretical framework from which we can construct and evaluate practical near-optimal solutions. In particular, *model-based* BRL approaches work by explicitly modelling the belief-state MDP; while *model-free* approaches, such as *Bayesian Q-learning*, attempt to learn action values directly, without solving an MDP. The following subsections discuss each of these in detail.

3.1 Model-Based BRL

In general, *model-based* BRL methods work by maintaining a density P over all possible dynamics D and reward functions R, which is updated with each observed tuple, $\langle s, a, r, s' \rangle$, where s' is the next state after action a is performed at s and reward r is received. This density describes the agent's belief state regarding the world, and is used to choose appropriate actions, given the current state of knowledge. Typically, updates are rendered tractable by assuming a convenient conjugate prior [8], which allows the belief state to be represented using a small set of *hyperparameters*, updated using a set of simple closed form equations.

For example, [6] models rewards and states as multinomial random variables, such that, for each s and a, a parameter set $\{\theta^{s'}_{s,a} | s' \in \mathcal{S}, \theta^{s'}_{s,a} = Pr(s'|s,a)\}$ defines the distribution of s' given s and a. In the same way, a similar set, $\{\theta^{r}_{s,a}\}$, models the conditional distribution over possible rewards. However, since these parameters are themselves unknown, each is assigned a conjugate prior, in this case a *Dirichlet*, specified by hyperparameters $\{\alpha^{s'}_{s,a}\}$ and $\{\alpha^{r}_{s,a}\}$. For example, if we observe a specific tuple $\langle s, a, r, s' \rangle$, the corresponding $\alpha^{s'}_{s,a}$ and $\alpha^{r}_{s,a}$ are both incremented by 1. In particular, if all hyperparameters are initialised to 1, this results in uniform densities, which become peaked around the true parameter values as more evidence is observed. In this way, the Dirichlets capture both the relative likelihood of possible multinomials, and the amount of uncertainty given the evidence. Given this, [6] proposes a tractable approximate solution to the belief-state MDP based on a myopic estimation of the expected Value of Perfect Information (VPI), defined by the expected *gain* in reward received, if the agent learns the *true* value of choosing action a in state s:

Definition 1 (VPI) *Let a_1 be an agent's current best action in state s, with expected Q-value m_{s,a_1}, and a_2 its 2nd best action, with expected Q-value m_{s,a_2}. According to [7], the* gain *for learning the true expected Q-value of an action, a, is then*

$$Gain_{s,a}(\mu_{s,a}) = \begin{cases} m_{s,a_2} - \mu_{s,a} & \text{if } a = a_1 \wedge \mu_{s,a} < m_{s,a_2}, \\ \mu_{s,a} - m_{s,a_1} & \text{if } a \neq a_1 \wedge \mu_{s,a} > m_{s,a_1}, \\ 0 & \text{otherwise.} \end{cases}$$

where $\mu_{s,a} = Q(s,a)$ is the true Q-value for a in s. Based on this, the value of perfect information (VPI) *for selecting a in s is defined as $VPI(s,a) = E[Gain_{s,a}(\mu_{s,a})]$.*

Intuitively, the gain reflects the effect on decision quality of learning the true $Q(s,a)$. In the first two cases, what is learned results in a change of decision: either because the estimated optimal action is found to be worse than predicted, or because some other action is found to be optimal. Otherwise, the information is irrelevant, since no change in decision is induced. Based on this, [6] proposes that an agent should choose actions that maximise $E[Q(s,a)] + VPI(s,a)$. In this way, exploration is encouraged by VPI when an agent is uncertain about its estimates, but the resulting policy approaches the optimal w.r.t the true Q-value, as VPI decreases in light of accumulated evidence.

3.2 Bayesian Q-Learning

The main problem with model-based BRL methods is that solving a belief-state MDP is generally intractable, and even approximate solutions can scale poorly in large problems. For example, in model-based BRL, VPI cannot be calculated analytically, but instead must be estimated by solving multiple MDPs sampled from an agent's belief state [6]. Fortunately, model-free techniques offer a simpler alternative, in which an agent directly learns the value for choosing an action in a given state, without explicitly solving an MDP. While this requires an agent to explore more to learn the true value of its actions (since the implications of observed evidence cannot be fully determined without modelling the MDP), the computational complexity of choosing an action is greatly reduced, which may be an important advantage in some on-line decision making problems.

In particular, standard Q-learning works by directly maintaining a point estimate of the Q-value, $Q(s,a)$, for each state and action, updated w.r.t. observed rewards. Unfortunately, it is *not* clear from this single estimate how much an agent should explore actions that are believed to be suboptimal, but may yet prove to be optimal.

In Bayesian Q-learning [7], this limitation is addressed by maintaining *a probability distribution* over $Q(s,a)$, which measures the uncertainty in the current estimate that can be used to guide exploration. More specifically, for each state-action pair, the total discounted reward is assumed to be normally distributed with unknown mean, $\mu_{s,a}$, and precision,[2] $\tau_{s,a} = 1/\sigma^2_{s,a}$, where $\sigma^2_{s,a}$ is the unknown variance of the distribution. Since $Q(s,a)$ is defined as the expected total discounted reward, we have $Q(s,a) = \mu_{s,a}$.

Now, to model the uncertainty in their estimates, Dearden et al. adopt the standard Bayesian approach of using conjugate parameter distributions for each pair of latent parameters, $(\mu_{s,a}, \tau_{s,a})$ [7, 8]. In this case, the joint distribution of $\mu_{s,a}$ and $\tau_{s,a}$ for each state-action pair is assumed to be a *normal-gamma* (*NG*) distribution, which is conjugate for normal densities with unknown mean and precision. More formally, we say that $(\mu_{s,a}, \tau_{s,a}) \sim NG(m_{s,a}, \lambda_{s,a}, \alpha_{s,a}, \beta_{s,a})$, where $\rho_{s,a} = \langle m_{s,a}, \lambda_{s,a}, \alpha_{s,a}, \beta_{s,a} \rangle$ are hyperparameters, updated according to the equations[3] in Theorem 1, which produce densities of the following form [8]:

$$p(\mu, \tau) \propto \tau^{\frac{1}{2}} e^{-\frac{1}{2}\lambda\tau(\mu-m)^2} \tau^{\alpha-1} e^{-\beta\tau} \quad (3)$$

Note that, in [7], the last term is *incorrectly stated* as $e^{\beta\tau}$, and so has the wrong sign within the exponent. Of course, we could always define a new hyperparameter $\hat{\beta} = -\beta$, and substitute this

[2] Here, the precision is used in place of the variance, because it simplifies the later Bayesian Analysis [8].

[3] In Bayesian Q-Learning, these update equations cannot be used directly because, although the latent distribution is over total discounted rewards, only *immediate* rewards can be directly observed. However, this technical detail [7] is not relevant to our discussion.

for β to correct the equation. However, in this case, the hyperparameter updates as stated in [7, 8] (Theorem 1) would also have to change, so in this sense, [7] is inconsistent.[4]

Theorem 1 (Posterior Hyperparameters) *Suppose that the prior density for the unknown parameters of a normal distribution is $p(\mu, \tau) = NG(m, \lambda, \alpha, \beta)$, and let $D = \{x_k\}_{k=1}^n$ be a set of n i.i.d. observations drawn from this distribution, with sample mean, $\bar{x} = \frac{1}{n}\sum_{k=1}^n x_k$, and sum of squares, $s^2 = \sum_{k=1}^n (x_k - \bar{x})^2$. As stated in [7, 8], the posterior is thus $p(\mu, \tau) \sim NG(m', \lambda', \alpha', \beta')$, with hyperparameters $\lambda' = \lambda + n$, $m' = (\lambda m + n\bar{x})/\lambda'$, $\alpha' = \alpha + n/2$ and $\beta' = \beta + s^2/2 + n\lambda(\bar{x} - m)^2/(2\lambda')$.*

From these *NG* distributions, a good estimate of $Q(s, a)$ can be obtained from $E[Q(s, a)] = E[\mu_{s,a}] = m_{s,a}$. However, in addition to such estimates, the parameter distributions also provide a representation of uncertainty. In particular, the marginal posterior distribution of $\mu_{s,a}$ generally becomes more peaked around its true value as more rewards are observed, and so the width of the distribution gives an indication of uncertainty. This can be used to guide principled exploration in Bayesian Reinforcement learning in a number of ways, the most promising of which is VPI action selection [7]. This works in the same way as described in Sec. 3 for model-based BRL, except that VPI can now be computed efficiently using a closed-form equation, without the need to sample and solve multiple MDPs. Unfortunately, the closed-form solution provided in [7] is inconsistent with the definition of VPI, and thus cannot be correct. In the next section, we highlight these inconsistencies in detail, and provide the *correct* analytical solution.

4. ANALYTICAL VPI FOR Q-LEARNING

As part of the Bayesian Q-Learning approach discussed above, [7] provides an *(incorrect)* analytical solution for the VPI, under the assumption that each $(\mu_{s,a}, \tau_{s,a})$ has a normal-gamma density with hyperparameters $m_{s,a}, \lambda_{s,a}, \alpha_{s,a}, \beta_{s,a}$. This result formed a critical part of the contribution of this paper, since without it, VPI would have to be calculated using numerical integration techniques, such as Monte Carlo sampling, thus introducing a significant computational overhead. In this section, we address this problem by (1) proving beyond doubt that the original equations are incorrect, and (2) providing the *correct* solution, which we prove in the appendix. With this in mind, we begin by restating the original result presented in [7], which we quote using the identity $E[\mu_{s,a}] = m_{s,a}$ for all s and a:

Proposition 1 (Dearden's Solution) *$VPI(s, a)$ is equal to $c + (m_{s,a_2} - m_{s,a_1})Pr(\mu_{s,a_1} < m_{s,a_2})$ when $a = a_1$, and $c + (m_{s,a} - m_{s,a_1}) \cdot Pr(\mu_{s,a} > m_{s,a_1})$ when $a \neq a_1$, where*

$$c = \frac{\Gamma\left(\alpha_{s,a} + \frac{1}{2}\right)\sqrt{\beta_{s,a}}}{\left(\alpha_{s,a} - \frac{1}{2}\right)\Gamma\left(\alpha_{s,a}\right)\Gamma\left(\frac{1}{2}\right)\sqrt{2\lambda_{s,a}}}\left(1 + \frac{m_{s,a}^2}{2\alpha_{s,a}}\right)^{-\alpha_{s,a}+\frac{1}{2}}$$

Here, the main problem is that c should *not* be constant w.r.t. m_{s,a_1} and m_{s,a_2}, but instead should depend on the difference between these two values and $m_{s,a}$. The following Lemma and Theorems show why this leads to inconsistent results.

Lemma 1 (Asymptotic Behaviour) *When $\alpha_{s,a} > \frac{1}{2}$, and $|m_{s,a}| \to \infty$, the term c from Proposition 1 goes to 0.*

PROOF. *If $\alpha_{s,a} > 1/2$ then $1/2 - \alpha_{s,a} < 0$. Therefore, since $\lim_{|m_{s,a}|\to\infty}(1 + m_{s,a}^2/2\alpha_{s,a}) = \infty$, $\lim_{|m_{s,a}|\to\infty} c = 0$.* □

[4]Theorem 1 is stated slightly differently in [7] and [8]. However, both are equivalent, differing only by some trivial transformations. Here, we follow the original reference [8] more closely.

Theorem 2 (Sensitivity to Value Changes) *If $d \in \mathbb{R}$, is added to all $\mu_{s,y}$ and $m_{s,y}$ for each action y, then this will change VPI according to Proposition 1. However, this is inconsistent with Definition 1, in which VPI is defined to be invariant to such changes.*

PROOF. *By Definition 1, if we add a constant, d, to $\mu_{s,a}$, m_{s,a_1}, and m_{s,a_2}, then $Gain_{s,a}(\mu_{s,a})$ and hence $VPI(s, a)$ remain unchanged. However, in Proposition 1, the term c depends only on $m_{s,a}$, and so it is sensitive to the addition of d, to which Definition 1 is invariant. In fact, in the limit $|d| \to \infty$ when $\alpha_{s,a} > 1/2$, $m_{s,a}$ also approaches ∞, and so from Lemma 1, c goes to zero.* □

Theorem 3 (Negative VPI) *By Proposition 1, VPI can be negative. However, this is inconsistent with Definition 1, in which VPI is strictly non-negative.*

PROOF. *Let $f(x, y) = (E[x] - y)\,Pr(x > y)$. By Definition 1, $Gain_{s,a}(\mu_{s,a}) \geq 0$, $\therefore VPI(s, a) = E[Gain_{s,a}(\mu_{s,a})] > 0$. In contrast, if $a \neq a_1 \wedge m_{s,a} < m_{s,a_1}$ then $f(\mu_{s,a}, m_{s,a_1})$ will be negative for non-zero $Pr(\mu_{s,a} > m_{s,a_1})$. However, if we add $d \in \mathbb{R}$, to $\mu_{s,a}$, $m_{s,a}$ and m_{s,a_1}, then $f(\mu_{s,a}, m_{s,a_1})$ will remain constant, while from Lemma 1, $c \to 0$ when $|d| \to \infty$. Therefore, according to Proposition 1, $VPI(s, a)$ will be negative for sufficiently large d, which is thus inconsistent with Definition 1. A similar argument can also be made when $a = a_1$.* □

From Theorems 2 and 3 it is clear that Proposition 1 cannot be true. As we shall show, however, the *correct* solution can be obtained by replacing Dearden et al.'s constant c with a *truncation bias* function, which we now define:

Definition 2 (Truncation Bias Function) *For hyperparameters $\rho = \langle m, \lambda, \alpha, \beta\rangle$, we define the* truncation bias function, *$\mathcal{B}_\rho : \mathbb{R} \to \mathbb{R}$, as follows.*

$$\mathcal{B}_\rho(x) = \frac{\Gamma\left(\alpha - \frac{1}{2}\right)\sqrt{\beta}\left(1 + \frac{\lambda(x-m)^2}{2\beta}\right)^{-\alpha+\frac{1}{2}}}{\Gamma(\alpha)\Gamma(1/2)\sqrt{2\lambda}}$$

Given this definition, the correct closed-form solution for VPI in Bayesian Q-Learning is given by Theorem 4:

Theorem 4 (VPI Solution) *According to an agent's beliefs, let a_1 be its current best action in state s, with expected reward m_{s,a_1}; and a_2 is its second best action, with expected reward m_{s,a_2}. Similarly, let a be an action whose reward in s is normally distributed, with unknown parameters $\langle \mu, \tau\rangle \sim NG(m, \lambda, \alpha, \beta)$, and hyperparameters $\rho = \langle m, \lambda, \alpha, \beta\rangle$. The VPI for choosing a in s is then $VPI(s, a) =$*

$$\begin{cases} (m_{s,a_2} - m) \cdot Pr(\mu | \mu < m_{s,a_2}) + \mathcal{B}_\rho(m_{s,a_2}) & \text{for } a = a_1 \\ (m - m_{s,a_1}) \cdot Pr(\mu | \mu > m_{s,a_1}) + \mathcal{B}_\rho(m_{s,a_1}) & \text{otherwise.} \end{cases}$$

As mentioned, this result is proved in the appendix, thus showing that it can be used to calculate VPI in Bayesian Q-learning, without the computational expense of numerical integration. In particular, we now introduce a general approach for decentralised BRL, including an efficient model-free algorithm based on this result.

5. DECENTRALIZED BAYESIAN RL

Sec. 2 described how certain multiagent MDPs can be decomposed into a set of regional reward and transition functions, and showed how this can be used to generate tractable solutions that approximate the optimal policy. However, this still assumes that the reward and transition functions are *known*, which is not the case in RL problems. As discussed in Sec. 3, the Bayesian RL approach deals with such cases by constructing and solving a belief state MDP, and so explicitly handles uncertainty over dynamics.

To put this in a multiagent MDP context, we define $\boldsymbol{b}$ as the joint belief state of all the agents, corresponding to some probability distribution over all possible models. More formally, $\boldsymbol{b}$ has the form $\boldsymbol{b} = \langle P_M; \boldsymbol{s} \rangle$, where P_M is some density over possible models (i.e., transition and reward dynamics); and $\boldsymbol{s}$ is the current state of the system (a vector of state variables). Given experience $\langle \boldsymbol{s}; \boldsymbol{a}; r; \boldsymbol{s}' \rangle$, where r is the observed global reward, $\boldsymbol{b}$ can be updated to $\boldsymbol{b}' = \boldsymbol{b}(\langle \boldsymbol{s}; \boldsymbol{a}; r; \boldsymbol{s}' \rangle) = \langle P'_M; \boldsymbol{s}' \rangle$ with updates given by Bayes rule (and implemented using standard Bayesian methods):

$$P'_M(m) = zPr(\boldsymbol{s}'; r | \boldsymbol{s}; \boldsymbol{a}; m) P_M(m) \tag{4}$$

where z is a normalising constant. Notice that the states and actions in the formulation above are *global* states and *joint* agent actions. However, by adopting a decomposition similar to that described in Sec. 2, we can formulate the problem based on local beliefs, in a way that facilitates tractable solutions. Specifically, we achieve this by making the following three assumptions. First, we assume that the global reward, r, can be factored into regional rewards, r_i, such that $r = \sum_i r_i$ over all regions, i. Second, we assume that the global belief state is decomposed into local beliefs states of the form $\boldsymbol{b}_i = \langle P_{M_i}; \boldsymbol{s}_i \rangle$, for each region i. These are updated as before, except that only local states, actions and rewards are observed:

$$\boldsymbol{b}'_i = \boldsymbol{b}_i(\langle \boldsymbol{s}_i; \boldsymbol{a}_i; r_i; \boldsymbol{s}'_i \rangle) = \langle P'_{M_i}; \boldsymbol{s}'_i \rangle$$

Finally, based on these two assumptions, we assume that the Q-value function can be factored as before, such that $Q(\boldsymbol{a}, \boldsymbol{b}) = \sum_i Q_i(\boldsymbol{a}_i, \boldsymbol{b}_i)$. While the addition of the first two assumptions may seem more restrictive than the factored MDP formulation in Sec. 2, the alternative is to assume global visibility of the full global state, joint actions and rewards. In fact, this is a more unrealistic assumption in coordination problems involving large numbers of distributed agents, so the assumptions above only make explicit what is already true in realistic settings.

Moreover, as we now show, these assumptions enable tractable coordinated reinforcement learning, in a way that explicitly accounts for uncertainty in the agents local beliefs, and so provide a near-optimal solution to the exploitation-exploration problem [18]. To achieve this, we propose a multiagent Bayesian RL method based on VPI, which consists of the following two steps. First, since (in general) no single agent has a complete view of the global problem, there is no global belief state from which to calculate VPI. Instead, within each region, the agents evaluate the informative value of performing a given local action w.r.t. the local belief state. Second, the agents coordinate their actions via message passing, in order to maximise the sum of all the regional expected Q-values and VPI. In this way, the agents not only coordinate their actions in a way that exploits the sum of their existing knowledge, but also *explore* joint actions that are informative for the regional belief states.

In detail, suppose that, given i's current belief state, the expected value of joint regional action $\boldsymbol{a}_i$ is given by $\overline{Q}_i(\boldsymbol{a}_i, \boldsymbol{s}_i)$. Letting $\boldsymbol{a}^1$ denote the regional action with highest expected q-value at $\boldsymbol{s}_i$ and $\boldsymbol{a}^2$ the second-highest, the regional VPI is defined as the *gain*, denoted $Gain_{\boldsymbol{a}_i, \boldsymbol{s}_i}(Q_i(\boldsymbol{a}_i, \boldsymbol{s}_i))$, from learning that the *true* Q_i value of taking $\boldsymbol{a}_i$ at $\boldsymbol{s}_i$ is in fact q:

$$Gain_{\boldsymbol{a}_i, \boldsymbol{s}_i}(q) = \begin{cases} \overline{Q}_i(\boldsymbol{a}^2, \boldsymbol{s}_i) - q, & \text{if } \boldsymbol{a}_i = \boldsymbol{a}^1 \wedge q < \overline{Q}_i(\boldsymbol{a}^2, \boldsymbol{s}_i) \\ q - \overline{Q}_i(\boldsymbol{a}^1, \boldsymbol{s}_i), & \text{if } \boldsymbol{a}_i \neq \boldsymbol{a}^1 \wedge q > \overline{Q}_i(\boldsymbol{a}^1, \boldsymbol{s}_i) \\ 0, & \text{otherwise} \end{cases}$$

The regional $VPI(\boldsymbol{a}_i, \boldsymbol{s}_i)$, defined as $E[Gain_{\boldsymbol{a}_i, \boldsymbol{s}_i}(Q_i(\boldsymbol{a}_i, \boldsymbol{s}_i))]$, is thus a direct analog of the standard notion of VPI, and can be added to the corresponding expected regional Q-value to boost the desirability of local actions with uncertain value. Thus, the *regional* value for taking the local joint action $\boldsymbol{a}_i$ in $\boldsymbol{s}_i$ can be defined as $\overline{Q}_i(\boldsymbol{a}_i, \boldsymbol{s}_i) + VPI(\boldsymbol{a}_i, \boldsymbol{s}_i)$. To use this definition for coordinated multiagent learning, we now propose two decentralised methods for BRL: (1) model-based decentralised BRL, which uses a distributed sampling approach to approximate the solution to the belief state MDP, and (2) decentralised Bayesian Q-learning, a model-free approach, which side-steps the computational complexity of solving the belief-state MDP, by learning the regional Q-values directly.

5.1 Decentralised Model-based BRL

As in standard model-based BRL (Sec. 3), model-based decentralised BRL works by sampling multiple MDPs, and using the solution to the MDPs to approximate the expected Q-value function, and the associated VPI. Specifically, we propose a four-step procedure:

1. For each region i, an agent representing i maintains a density, P_{M_i}, over all possible local transition and reward dynamics, updated using local observations only. For example, as used in our experiments (Sec. 6) this may be achieved by (1) modelling local state transition probabilities, $Pr(\boldsymbol{s}'_i | \boldsymbol{s}_i, \boldsymbol{a}_i)$, as a set of unknown multinomial distributions with associated Dirichlet priors; and (2) modelling local reward distributions, $Pr(r_i | \boldsymbol{s}_i, \boldsymbol{a}_i)$, as unknown Gaussians with associated normal-gamma priors.[5]

2. In each region, the representative agent samples a finite set of z local (reward and transition) models from the corresponding density, P_{M_i}; that is, z samples for every $i \in [1, Y]$ are specified ($z * Y$ in total). These are used to form a set of z distinct factored MDPs, such that the kth factored MDP comprises the kth local reward and transition functions sampled from each of the Y regions. Each of these factored MDPs represents one possible instance of the joint decision problem, which are solved to produce the set of local $Q_i(\boldsymbol{a}_i, \boldsymbol{s}_i)$ values, for the corresponding joint optimal policy. In this paper, we achieve this using our own decentralised dynamic programming algorithm (not described here) based on max-sum. However, this choice does not significantly change the end result, and so may be replaced by any suitable algorithm for factored MDPs (e.g. [10]).

3. For each region, we calculate the average $Q_i(\boldsymbol{a}_i, \boldsymbol{s}_i)$ from the z sampled MDPs, and use this to approximate $\overline{Q}_i(\boldsymbol{a}_i, \boldsymbol{s}_i)$. Similarly, we compute $Gain_{\boldsymbol{a}_i, \boldsymbol{s}_i}(Q_i(\boldsymbol{a}_i, \boldsymbol{s}_i))$ for each of the z MDPs w.r.t. i, and approximate $VPI(\boldsymbol{a}_i, \boldsymbol{s}_i)$ by their average.

4. The local value of $\boldsymbol{a}_i$ (for region i) is defined to be $\overline{Q}_i(\boldsymbol{a}_i, \boldsymbol{s}_i) + VPI(\boldsymbol{a}_i, \boldsymbol{s}_i)$; $\boldsymbol{a}_i$'s desirability is thus boosted by its expected VPI. When the agents come to act, these are then evaluated w.r.t. *current* state to form the factors of a factor graph that can be operated on by the standard max-sum algorithm. The max-sum output at each variable node (one per agent j) gives the action choice for j. As a consequence, each agent's decision is informed by the entire global state through its affect on local rewards.

Although this procedure does not guarantee an optimal solution to the generally intractable decision problem, it does provide a practical alternative that maintains several useful features of the theoretical optimum. In particular, the look-ahead performed by solving the factored MDPs takes into account the likely impact of each agent's current actions on the future rewards obtained by the system as a whole. Moreover, by employing VPI, we explicitly account for

[5]This differs from [6], in which rewards are assumed to be multinomial rather than normally distributed. However, both approaches are valid, the first being appropriate when rewards are drawn from a known finite set, while the latter allows for any real value.

the exploratory value of each joint action for obtaining relevant information about regional rewards. In this way, agents will only explore joint actions that are likely to produce beneficial gains in their future rewards. Equally important, however, is the scalability of the procedure, which it achieves through the use of max-sum and factored MDPs (Sec. 2).

5.2 Decentralised Model-Free BRL

Although the above procedure scales well to problems involving large numbers of agents, sampling and solving multiple factored MDPs may still present a significant overhead when computational resources are at a premium, such as in sensor networks or UAVs with embedded CPUs. As we have already seen however, this problem can be side-stepped using model-free methods that attempt to learn the Q-value functions directly. With this in mind, we now adapt Bayesian Q-learning for decentralised settings, by modifying the model-based procedure above in the following way.

1. Rather than maintain a density over all possible transition and reward dynamics, each P_{M_i} now becomes a normal-gamma (*NG*) density, which *directly* models the distribution over all possible regional Q-value functions. This density is maintained and updated using the same procedures proposed in [7], except we now maintain separate models for each regional Q-value, rather than a single global one.
2. Rather than approximate the regional value functions by sampling, $E[Q_i(\boldsymbol{a}_i, \boldsymbol{s}_i)]$ is given directly by the mean of the corresponding *NG* distribution, and $VPI(\boldsymbol{a}_i, \boldsymbol{s}_i)$ can be calculated analytically using our closed-form solution in Sec. 4. As before, by summing these two values together for each region, we obtain a factor graph that can be operated on directly by max-sum to coordinate the agents' actions w.r.t. the current global state.

This procedure is similar to the decentralised Q-learning algorithm in [13], except that by adopting a Bayesian approach, we perform more efficient exploration of the state-action space. This ability is demonstrated empirically in the next section, using the algorithm in [13] as a benchmark.

6. EMPIRICAL EVALUATION

We now evaluate our proposed decentralised BRL methods by simulating the scenario in Fig. 1. Here, two factors may influence performance: (1) the priors required by each algorithm, and (2) the complexity of the task being learnt.

To investigate the former, we ran multiple simulations using priors with varying hyperparameter values. In particular, as suggested in Sec. 5.1, we used Dirichlet priors to model the regional state transition probabilities used by our model-based algorithm, along with normal-gamma priors for the regional rewards. Similarly, for our decentralised Bayesian Q-learning algorithm, we used normal-gamma priors to directly model the regional Q-value distributions, thus avoiding the need for separate transition and reward models.

To investigate the latter, we simulated two variants of the Fig. 1 scenario. In the first variant, we simulated the scenario exactly as described in Fig. 1, with 6 UAVs bordering on 5 regions with 1 target. Specifically, each UAV has a base station where it can land during idle periods, and is responsible for patrolling regions adjacent to its base station, east and west along the road. When a vehicle is detected in a region (e.g. by ground based motion sensors), the pair of UAVs bordering the region are alerted, and, importantly, must *both* patrol the region *simultaneously* to observe the vehicle. Since UAVs 1 and 6 border only one region each, their actions are limited to remaining idle and patrolling east or west respectively. All other UAVs can patrol both east and west, or remain idle. A region incurs a cost of -1 if one of its UAVs is active (regardless of direction), -2 if both are active, and receives a reward of 30 after every 3 observations.[6] Each region only communicates directly with its immediate neighbouring regions, and is only aware of its two bordering UAVs' actions. The number of target observations is visible to all regions, but its location is known only to its current region, and those immediately east and west. In the second variant, the state-action space complexity is reduced by decreasing the number of regions to 3, while at the same, the lookahead required is increased, by changing the number of observations required to receive a positive reward of 30 from 3 observations to 4.

The combined size of the local state and action spaces in the first variant is thus 54 for regions *a* and *b*, and 108 elsewhere; compared to 4860 for the global problem. This illustrates the reductive power of decomposition to simplify the combinatorial problem faced by the agents, thus turning a potentially intractable problem into a solvable one. Despite this simplification, we can still learn an effective policy for the global problem, as we now demonstrate.

In each scenario variant, we benchmark against two other strategies: (1) *random*, which selects actions with equal likelihood, and thus represents a basic solution that any algorithm should outperform; and (2) *Kok*, the decentralized Q-learning policy proposed in [13] (named after the lead author). The latter maintains separate estimates for each regional Q-value, and is the only other algorithm in the literature that uses max-sum decentralised reinforcement learning. However, unlike our model-free method, this maintains point estimates of the Q-values only, and uses ϵ-greedy exploration with a fixed exploration probability of $\epsilon = 0.2$ (see [13] for details).

Fig. 2 plots the mean cumulative rewards at each timestep of these experiments, calculated using $\approx$ 100 independent runs per control condition for statistical significance. As we discuss below, the most interesting effect induced by the choice of prior can be observed when the normal-gamma λ hyperparameter is varied, while all other hyperparameters remain constant. For this reason, we focus on λ in this discussion. In particular, Fig. 2(a) and (b), show the results for our model-based algorithm (labelled *MB*) in the 3 and 5 region problems respectively, when the λ hyperparameter of the prior distribution over rewards was varied in the range $[10^{-2}, 10^{-5}]$. In each case, a uniform Dirichlet prior was used for the state transition probabilities, while the other hyperparameters for the rewards where initialised to $\mu = 0$, $\alpha = 1$, $\beta = 1$. There are two main results of these experiments.

First, for all hyperparameter values tested, our model-based approach outperforms both the random strategy, and the *Kok* algorithm. This is because our model-based performs targeted exploration early on, taking account of its initial uncertainty. In contrast, although *Kok* learns quickly to prefer idle states that cost nothing (allowing it to dominate at the beginning) it takes significantly longer to learn that positive rewards can be achieved by coordinated observations of the target. In fact, in the harder 5 region problem (Fig. 2 b), *Kok* fails to learn how to observe the target at all within 4000 timesteps, a result which is backed by experiments in [13], which required >10,000 episodes to learn in a similar setting. Thus, although our model-based approach incurs an initial cost by performing early exploration, this enables the UAVs to learn how to coordinate their observations in significantly fewer timesteps than *Kok*.[7]

[6] Here, by requiring UAVs to perform multiple observations, we are able to evaluate our algorithms' ability to learn non-myopic policies. The UAVs must learn to balance the immediate cost of observing the target, against the expected gain in future reward.

[7] Although the learning time may seem large, the no. states & actions is equally large, and strategies start with uninformative priors.

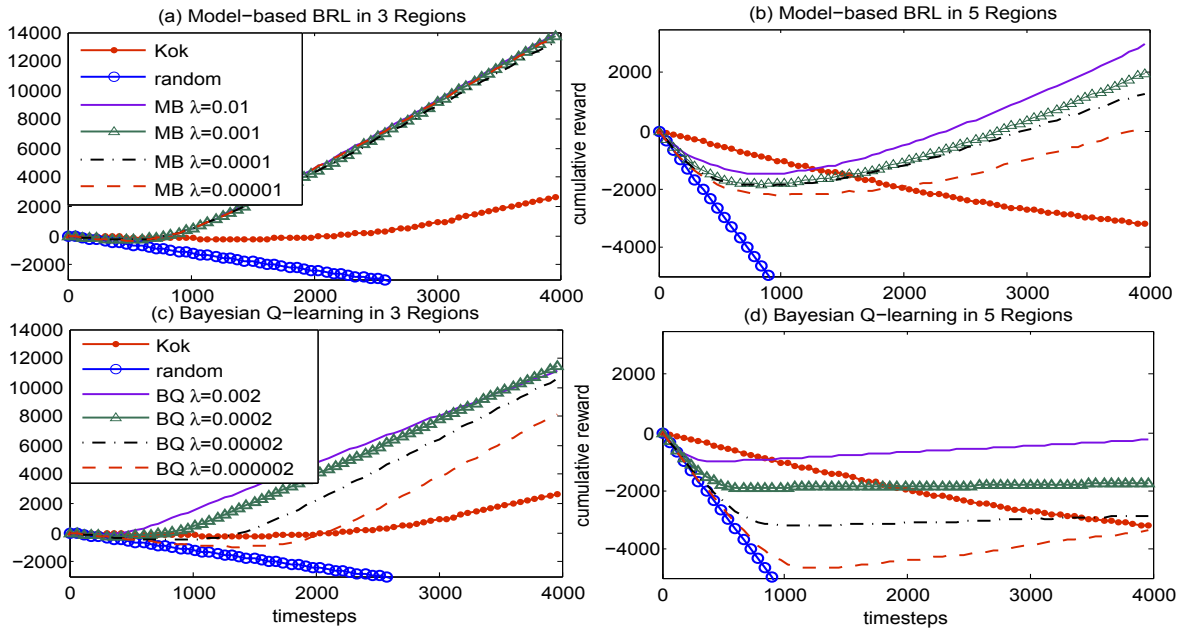


Figure 2: UAV Surveillance Scenario Results

Second, although our model-based algorithm performs well generally, changing the prior can have a significant effect on performance, for example, when the λ hyperparameter of the reward's NG prior is varied in the 5-region problem (Fig. 2 b).[8] In general, this is to be expected, because strong prior information will always bias inference in a certain way. Nevertheless, at first sight, the results here are somewhat surprising, since in these experiments, we use supposedly uninformative priors, which should be quickly dominated by observed evidence. Closer inspection provides two reasons for this result. First, since the state-action space is relatively large, the prior's effect can persist over parts of the MDP, because of the time required to fully explore all state-action pairs. Second, the range of rewards considered probable can significantly effect the amount of exploration performed. In particular, although changes in $\lambda \ll 1$ have little impact on posterior NG distributions, *a priori*, each decrease in λ by a factor of 10 produces an equivalent increase in the range of probable rewards. As a result, agents become more optimistic about the value of potential rewards, and so are incentivised to explore otherwise suboptimal policies, on the chance that they may (even occasionally) return very high rewards. While this dependence on priors may seem like a disadvantage, it should be noted that non-Bayesian approaches usually rely on tuning parameters with less obvious interpretations. In contrast, prior distributions do have an intuitive interpretation, and in most domains, the range of likely rewards is known *a priori*.

As shown in Fig. 2(c) and (d), our model-free approach achieves similar results, except for a greater dependence on the correct choice of lambda.[9] This is because, without explicitly modelling the underlying MDP, it cannot infer the full consequences of its observations, and so requires more exploration to rule out occasionally high rewards from the full range of policies. As such, although the model-based algorithm has a higher computational complexity (see below), it can learn effectively from less evidence. This may be particularly advantageous in robotics, where the cost of performing actions with real hardware may outweigh the additional computational overhead. In this sense, even our model-free approach significantly outperforms *Kok*, making it a useful compromise in domains requiring both computational and learning efficiency.

In terms of time complexity, it is true that our methods take longer to choose actions compared to the simpler *Kok* approach. For example, using our implementation (which leaves significant room for optimization), it took the model-based learner on average 11 ± 6 secs. to choose each action, compared to 0.2 ± 0.1 secs for the Bayesian Q-Learner, and 0.04 ± 0.02 secs. for the *Kok* approach. However, notice that our approaches outperform other methods in terms of the *timesteps* required to learn. This is important in many real-world domains that use real hardware (e.g. UAVs), where repeated interactions with the environment may be time consuming, costly, or potentially dangerous. In such domains, it makes sense to deliberate over each action for longer to save time and resources in the long run; as fewer, as opposed to more, interactions for learning are strongly preferred. BRL is ideally suited to this in general; and by exploiting regional decomposition, our approach can address otherwise intractable coordination problems.

7. CONCLUSIONS AND FUTURE WORK

In this paper, we presented the *first* approach for performing cooperative multiagent BRL; and provide the *correct* closed form equations for VPI in Bayesian Q-Learning [7] — a crucial result with implications *beyond* our decentralised setting. Key to our approach is the use of factored MDPs, which significantly reduce complexity in structured coordination problems. In this sense, our experiments are somewhat preliminary, since factored MDPs can be applied to problems larger than those attempted here [10]. Nevertheless, our results still demonstrate the potential of BRL to outperform existing multiagent learning algorithms, and so, in future work, we plan to evaluate our approach in larger problems, by taking advantage of advances in related areas, such as Monte-Carlo Planning [17] and ND-POMDPs [16].

Typically, informative priors significantly reduce learning times.

[8]This and all other claims made here are verified by t-tests with a confidence level of at least 95%.

[9]In the model-free experiments λ was scaled by 0.2 due to the change from modelling immediate rewards to Q-values.

8. REFERENCES

[1] S. M. Aji and R. J. McEliece. The generalized distributive law. *Information Theory, IEEE Transactions on*, 46(2):325–343, 2000.

[2] D. S. Bernstein, S. Zilberstein, and N. Immerman. The complexity of decentralized control of markov decision processes. In *Proc. of UAI-2000*, pages 32–37, 2000.

[3] C. Boutilier. Sequential optimality and coordination in multiagent systems. In *Proc. of IJCAI-99*, pages 478–485, 1999.

[4] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, 11:1–94, 1999.

[5] G. Chalkiadakis and C. Boutilier. Sequentially optimal repeated coalition formation under uncertainty. *Autonomous Agents and Multi-Agent Systems*, 24(3):441–484, 2012.

[6] R. Dearden, N. Friedman, and D. Andre. Model based bayesian exploration. In *Proc. of UAI'99*, 1999.

[7] R. Dearden, N. Friedman, and S. Russell. Bayesian Q-Learning. In *Proc. of AAAI-98*, 1998.

[8] M. DeGroot and M. Schervish. *Probability & Statistics*. Pearson Education, 3rd edition, 2002.

[9] 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 AAMAS 2008*, pages 639–646, 2008.

[10] C. Guestrin, D. Koller, and R. Parr. Max-norm projections for factored mdps. In *Proc. of AAAI-01*, pages 673–680, 2001.

[11] C. Guestrin, M. Lagoudakis, and R. Parr. Coordinated reinforcement learning. In *Proc. of ICML-02*, pages 227–234, 2002.

[12] H. J. Kim. Moments of truncated student-t distribution. *Journal of the Korean Statistical Society*, 37:81–87, 2008.

[13] J. R. Kok and N. Vlassis. Collaborative multiagent reinforcement learning by payoff propagation. *Journal of Machine Learning Research*, 7:1789–1828, 2006.

[14] F. R. Kschischang, B. J. Frey, and H. A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Trans. on Information Theory*, 42(2):498–519, 2001.

[15] J. Martin. *Bayesian decision problems and Markov chains*. Wiley, 1967.

[16] R. Nair, P. Varakantham, M. Tambe, and M. Yokoo. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *Proceedings of AAAI-05*, pages 133–139. 2005.

[17] D. Silver and J. Veness. Monte-carlo planning in large pomdps. In *Neural Information Processing Systems 23*, pages 2164–2172. 2010.

[18] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[19] Y. Weiss and W. T. Freeman. On the optimality of solutions of the max-product belief propagation algorithm in arbitrary graphs. *IEEE Trans. on Information Theory*, 47(2):723–735, 2001.

Acknowledgments

This research was funded by the UK Engineering and Physical Sciences Research Council (EPSRC) as part of the SUAAVE and ORCHID projects (grant references EP/F06358X/1 and EP/I011587/1). Georgios Chalkiadakis has been partially supported by the European Commission FP7-ICT Cognitive Systems, Interaction, and Robotics under the contract #270180 (NOPTILUS).

APPENDIX

We now prove that Theorem 4 provides the *correct* solution for VPI in Bayesian Q-Learning, which [7] states incorrectly. We start with the case when $a = a_1$, and for simplicity, drop the subscripts for the hyperparameters of a, so that $\mu = \mu_{s,a}$ and so on. Then from Definition 1, we have $VPI(s,a) = E[m_{s,a_2} - \mu]$ given $\mu < m_{s,a_2}$, and 0 otherwise. However, since the truth of $\mu < m_{s,a_2}$ is unknown, we must marginalise to derive the correct expectation:

$$VPI(s,a) = (m_{s,a_2} - E[\mu|\mu < m_{s,a_2}])\, Pr(\mu < m_{s,a_2})$$

Similarly, when $a \neq a_1$, we find that

$$VPI(s,a) = (E[\mu|\mu > m_{s,a_1}] - m_{s,a_1})\, Pr(\mu > m_{s,a_1})$$

Therefore, to prove Theorem 4, we need only show that

$$\begin{aligned} \forall x \in \mathbb{R} \quad E[\mu|\mu < x]Pr(\mu < x) &= m \cdot Pr(\mu < x) - \mathcal{B}_\rho(x) \\ \wedge \quad E[\mu|\mu > x]Pr(\mu > x) &= m \cdot Pr(\mu > x) + \mathcal{B}_\rho(x) \end{aligned}$$

To achieve this, we first state some prerequisite results, which are then used to prove these equations in Lemma 5.

Lemma 2 *Let X and Y be continuous random variables such that the c.d.f. of X is $F(x)$ and $Y = \sigma X + \mu$. By substitution [8], $Pr(Y < y) = F\left[(y-\mu)/\sigma\right]$, where $x = (y-\mu)/\sigma$, and so $Pr(Y > y) = 1 - F\left[(y-\mu)/\sigma\right]$.*

Lemma 3 *Suppose that X is t-distributed with v degrees of freedom, and $F_v(\cdot)$ its c.d.f. From [12], when $X \in (a,b)$ is given, its expected value is*

$$E[X|a < X < b] = \frac{\Gamma\left(\frac{v-1}{2}\right) v^{v/2} \left(A_{(v)}^{-(v-1)/2} - B_{(v)}^{-(v-1)/2}\right)}{2\left[F_v(b) - F_v(a)\right]\Gamma(v/2)\Gamma(1/2)}$$

for $v > 1$, where $A_{(v)} = v + a^2$ and $B_{(v)} = v + b^2$.

Corollary 1 *By taking the limits $a \to -\infty$ and $b \to \infty$ respectively, when $v > 1$, it follows from Lemma 3 that*

$$\begin{aligned} E[X|X < b] &= -\frac{\Gamma\left(\frac{v-1}{2}\right) v^{v/2} \left(v + b^2\right)^{-(v-1)/2}}{2F_v(b)\Gamma(v/2)\Gamma(1/2)} \\ E[X|X > a] &= \frac{\Gamma\left(\frac{v-1}{2}\right) v^{v/2} \left(v + a^2\right)^{-(v-1)/2}}{2\left[1 - F_v(a)\right]\Gamma(v/2)\Gamma(1/2)} \end{aligned}$$

Lemma 4 *If $\langle \mu, \tau \rangle \sim NG(m, \lambda, \alpha, \beta)$ are the unknown parameters of a normal distribution, then $Z = (\mu - m)\sqrt{\lambda\alpha/\beta}$ is t-distributed with 2α degrees of freedom [8], from Lemmas 2 & 3, we thus have $Pr(\mu < y) = F_{2\alpha}\left[(y-m)\sqrt{\lambda\alpha/\beta}\right]$.*

Lemma 5 *If $\langle \mu, \tau \rangle \sim NG(m, \lambda, \alpha, \beta)$ are the unknown parameters of a normal p.d.f. and $\rho = \langle m, \lambda, \alpha, \beta \rangle$, then*

$$E[\mu|\mu < x]Pr(\mu < x) = m \cdot Pr(\mu < x) - \mathcal{B}_\rho(x) \quad (5)$$

$$E[\mu|\mu > x]Pr(\mu > x) = m \cdot Pr(\mu > x) + \mathcal{B}_\rho(x) \quad (6)$$

PROOF. *If $Z = (\mu - m)\sqrt{\lambda\alpha/\beta}$ and $y = (x-m)\sqrt{\lambda\alpha/\beta}$ then $Pr(\mu < x) = Pr(Z < y)$, $Pr(\mu > x) = Pr(Z > y)$, and*

$$E[\mu|\mu < x] = m + \sqrt{\beta/\lambda\alpha} \cdot E[Z|Z < y] \quad (7)$$

$$E[\mu|\mu > x] = m + \sqrt{\beta/\lambda\alpha} \cdot E[Z|Z > y] \quad (8)$$

From Lemma 4, Z is t-distributed with $v = 2\alpha$ degrees of freedom, and so $\forall y \in \mathbb{R}, 0 < Pr(Z < y) = F_v(y) < 1$. Thus, by substitution into Corollary 1 we have

$$\begin{aligned} E[Z|Z < y] &= -\frac{\Gamma\left(\alpha - \frac{1}{2}\right)(2\alpha)^\alpha \left(2\alpha + \frac{\lambda\alpha(x-m)^2}{\beta}\right)^{-\alpha+\frac{1}{2}}}{2Pr(Z < y)\Gamma(\alpha)\Gamma(1/2)} \\ E[Z|Z < y] &= -\frac{\Gamma\left(\alpha - \frac{1}{2}\right)\sqrt{\alpha}\left(1 + \frac{\lambda(x-m)^2}{2\beta}\right)^{-\alpha+\frac{1}{2}}}{\sqrt{2}Pr(Z < y)\Gamma(\alpha)\Gamma(1/2)} \\ E[Z|Z < y] &= -\sqrt{\lambda\alpha/\beta} \cdot \mathcal{B}_\rho(x)/Pr(Z < y) \quad (9) \end{aligned}$$

By following the same procedure for $\mu > x$, we obtain

$$E[Z|Z > y] = \sqrt{\lambda\alpha/\beta} \cdot \mathcal{B}_\rho(x)/Pr(Z > y) \quad (10)$$

By substituting Eqs. 9 & 10 into Eqs. 7 & 8, we obtain

$$E[\mu|\mu < x] = m - \mathcal{B}_\rho(x)/Pr(\mu < x) \quad (11)$$

$$E[\mu|\mu > x] = m + \mathcal{B}_\rho(x)/Pr(\mu > x) \quad (12)$$

From this, Eqs. 5 & 6 follow directly, thus proving the lemma, and proving Theorem 4 as a consequence. □

Shaping Fitness Functions for Coevolving Cooperative Multiagent Systems

Mitchell Colby
Oregon State University
442 Rogers Hall
Corvallis, OR 97331
colbym@engr.orst.edu

Kagan Tumer
Oregon State University
204 Rogers Hall
Corvallis, OR 97331
kagan.tumer@oregonstate.edu

ABSTRACT

Coevolution is a natural approach to evolve teams of agents which must cooperate to achieve some system objective. However, in many coevolutionary approaches, credit assignment is often subjective and context dependent, as the fitness of an individual agent strongly depends on the actions of the agents with which it collaborates. In order to alleviate this problem, we introduce a cooperative coevolutionary algorithm which biases the evolutionary search as well as shapes agent fitness functions to reward behavior that benefits the system. More specifically, we bias the search using a hall of fame approximation of optimal collaborators, and we shape the agent fitness using the difference evaluation function. Our results show that shaping agent fitness with the difference evaluation improves system performance by up to 50%, and adding an additional fitness bias can improve performance by up to 75%.

Categories and Subject Descriptors

H.3.4 [**Systems and Software**]: Distributed Systems

General Terms

Algorithms, Experimentation

Keywords

Co-evolution, Multiagent learning

1. INTRODUCTION

Coordinating multiple agents in order to achieve some system objective is an important area of research, and is critical in many domains including rover coordinaton, air traffic control, search and rescue, and unmanned aerial vehicle coordination[1, 2]. One approach to achieving coordination is the use of Cooperative Coevolutionary Algorithms (CCEAs), which involve evolving multiple populations simultaneously and evaluating the fitness of individuals based on the individual's interactions with other agents in the system [10]. By evolving multiple populations at once, CCEAs project the search space into multiple, smaller, search spaces. Coevolving agents have only a fraction of the total state space available to them. Further, agents' fitness assignments are context dependent and are influenced by agents from other populations. Thus, CCEAs have the tendency to create agents which are capable of performing adequately with a wide range of collaborators, rather than specializing to perform well with the best set of collaborators; in other words, CCEAs often produce stable, rather than optimal solutions [3, 4, 10]. In order to make CCEAs a viable option for coordinating multiagent systems, it is critical that steps be taken to achieve optimal coordination policies.

There have been multiple approaches to address the issues of suboptimal stable policies. One approach involves shaping local fitness functions to align with the system evaluation function. Proper shaping of these fitness functions leads to faster learning and more optimal policies [1, 8]. Another approach is to bias searches based on the notion of optimal teammates [10], which involves estimating agent utilities as if they were paired with optimal collaborators. Although these biased searches generally increase the effectiveness of CCEAs, an issue with this approach is that in complex domains, estimating system evaluations as if optimal collaborators were present is exceedingly difficult. Other methods involve altering the evolutionary mechanisms in CEAs in order to optimize CEA performance, such as lenient learners or hall of fame methods [11, 13].

In this work we address the suboptimal solutions created by CCEAs by shaping fitness functions and approximating optimal collaborators. Two domains are used to test whether these problems are addressed adequately. First, a *scatter domain*, which involves agents moving in a two dimensional plane in order to become as "spread out" as possible. Secondly, a *rover domain* involving robots gathering data from points of interest tested the algorithms on a more real-world problem.

The contribution of this paper is a CCEA which:

- Shapes fitness functions using the Difference Utility
- Biases search with optimal collaborators using a hall of fame approach, which is much less complex than approximating optimal collaborators in an ad-hoc manner

In our experiments, this algorithm outperformed a CCEA using the system evaluation to assign fitness by an average of 48.7%. The remainder of this paper is organized as follows: section 2 describes related work and background information. Section 3 describes the domains analyzed. Section 4 describes the algorithms used in this research. Section 5

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

gives the experimental results. Finally, Section 6 discusses the results and potential areas for future research.

2. EVOLUTION AND COEVOLUTION

Evolutionary Algorithms (EAs) are a class of stochastic search algorithms which often outperform classical optimization techniques, particularly in complex domains where gradient information is not available [6]. An evolutionary algorithm typically contains three basic mechanisms: solution generation, mutation, and selection. These mechanisms are used on an initial set of candidate solutions, or a population to generate new solutions and retain solutions that show improvement. Simple EAs are excellent tools, but need to be modified to be applicable to large multiagent search problems. One such modification is *coevolution*, where multiple populations evolve simultaneously in order to develop policies for interacting agents. The following sections introduce coevolutionary algorithms, and approaches to optimize the output of these algorithms.

Coevolutionary Algorithms (CEAs) are an extension of evolutionary algorithms and are often well-suited for multiagent domains [5]. In a CEA, the fitness of an individual is based on its interactions with other agents it collaborates with. Thus, assessing the fitness of each agent is context-sensitive and subjective [10]. In *competitive* coevolution, individuals benefit when other agents fail. In *cooperative* coevolution, individuals succeed or fail as a team. This paper is focused on Cooperative Coevolutionary Algorithms (CCEAs). One of the advantages to coevolution is that the algorithm only needs to search subspaces of the state space, rather than the entire state space. This reduced state space often makes the learning process simpler for the cooperating agents. However, these simpler subspaces represent a large loss in information; the consequence of this is that the policies obtained by using these state projections are strongly influenced by other populations. The result is that agents evolve to partner well with a broad range of other agents, rather than evolving to form optimal partnerships [10]. Thus, in addition to trying to decrease the complexity of the learning process, research in coevolution aims to achieve optimal policies rather than stable ones.

Cooperative Coevolutionary Algorithms.

CCEAs are a natural approach in domains where agents succeed or fail as a team [12]. In CCEAs, distinct populations evolve simultaneously, and agents from these populations collaborate to reach good system solutions. One issue with CCEAs is that they tend to favor stable solutions, rather than optimal solutions [16]. This phenomena occurs because the different evolving populations adapt to each other, rather than adapting to form an optimal policy. Another issue that arises with CCEAs is the problem of credit assignment. Since the agents succeed or fail as a team, the fitness of each agent becomes subjective and context-dependent (e.g. an agent might be a "good" agent, but the agents it collaborates with are "bad," and the objective isn't reached. In this case, the "good" agent may be perceived as "bad") [16]. Generally speaking, research in CCEAs involves either making a more computationally efficient algorithm or reaching better solutions (avoiding suboptimal equilibria) [1]. The following sections outline some methods that have been developed to mitigate problems associated with CCEAs.

Biasing Coevolutionary Search

Panait *et al.*[10] biased the evolutionary search in order to find optimal solutions, rather than becoming stuck in local minima. In standard coevolution, a single agent is rated on how well the team does as a whole; every agent in the team gets equal credit. This results in an unfavorable signal-to-noise ratio, as each agent is unaware of its individual contribution to the team's performance. In a *Biased Cooperative Coevolutionary Algorithm* (BCCEA), the fitness of an individual is based partly on its interactions with other agents (as in usual CCEAs), and partly on an estimate of the best possible fitness for that individual if it is partnered with optimal collaborators. By biasing the coevolutionary search in this manner, the algorithm is better able to search for optimal policies, rather than stable policies, because each agent receives feedback related to its performance, rather than solely the team's performance. One issue with this approach is that estimating how an agent would perform with optimal collaborators is a nontrivial task, and becomes increasingly difficult in complex domains.

Fitness Function Shaping

Hoen and De Jong shaped the utilities of the agents so as to contribute to the system evaluation, such that an agent maximizing its individual utility would act to increase the system evaluation. By shaping agent fitnesses, the learning process is sped up considerably [8]. This work is similar to that of Agogino and Tumer, and Knudson and Tumer, who utilized difference evaluations as fitness functions to evolve coordination in multiagent systems [1, 9]. The *Difference Evaluation* is defined as:

$$D_i \equiv G(z) - G(z_{-i} + c_i) \tag{1}$$

where $G(z)$ is the total system evaluation, z_{-i} are all the states on which agent i has no effect, and c_i is the *counterfactual* term, which is a fixed vector to replace the effects of agent i. Intuitively, the second term in Equation 1 evaluates the fitness of the system without the effects of agent i, so the difference evaluation gives agent i's contribution to the system evaluation [1]. By shaping fitness functions such that each agent's fitness is related to the individual's contribution to team performance, the signal-to-noise ratio is improved considerably.

Evolving Teams

Coevolution is frequently a good algorithm to use in multiagent systems with heterogeneous agents. Haynes *et al.* used genetic algorithms to evolve *teams* of predators [7]. Rather than evolving single predators, one member of the population consisted of four predators. In this manner, the predators can evolve to cooperate. By evolving teams rather than individual agents, communication is not required in the domain; in place of communication, the team of predators evolves to act as if they know the other agents' future actions based on the state of the system [15] Though effective, these approaches are designed for small teams, and become slow to converge when scaled to large multiagent systems.

Hall of Fame

Rosin and Belew[13] introduced the concept of the *Hall of Fame* for competitive coevolution, in which top individuals are saved in order to test against in later generations. There

are two reasons why it is beneficial to save these top individuals. First, keeping top individuals contributes genetic information to later generations, which is imperative when conducting any evolutionary algorithm. Secondly, by keeping top individuals, new individuals in later generations may be tested against the hall of fame members. This concept can be extended to CCEAs by keeping hall of fame *teams*, rather than hall of fame *individuals*. In this manner, desirable genotypes won't be lost if they perform poorly for a few generations due to stochasticity.

Leniency

Panait *et al.*[11] introduced the concept of *lenient* learners in coevolutionary algorithms, which are agents which forgive possible mismatched teammate actions that result in poor team performance. Using lenient learners in coevolution is shown to provide learners with more accurate information about their policies, which increases the likelihood of converging to an optimal solution. Leniency is achieved by pairing agents with multiple sets of collaborators, and taking the highest team fitness achieved from all runs. This lowers the likelihood that a learning agent will receive poor feedback simply because it was paired with suboptimal teammates.

3. EVALUATION DOMAINS

In this section, we introduce the two problems analyzed in this work, and provide a detailed explanation of the system dynamics and evaluation functions used in each domain.

3.1 Scatter Domain

The scatter domain used in this paper is a variant of the mixing problem [14]. In the scatter domain, a team of agents on a two dimensional plane aim to move around and configure themselves to be as "spread out" as possible (Figure 1). The world is continuous, as are the actions of each agent. Each agent calculates the distance between itself and the closest teammate using the standard Euclidian distance. So, if there are N agents, each agent calculates how far away the closest agent is at any time t using:

$$\delta_i(t) = \min_j \left\{ \sqrt{(x_{i,t} - x_{j,t})^2 + (y_{i,t} - y_{j,t})^2} \middle| i \neq j \right\} \quad (2)$$

where $\{x_{i,t}, y_{i,t}\}$ is agent i's x and y position in the world at time t, and j is used to index all agents other than agent i. The total state of the system z is the set of all agent positions

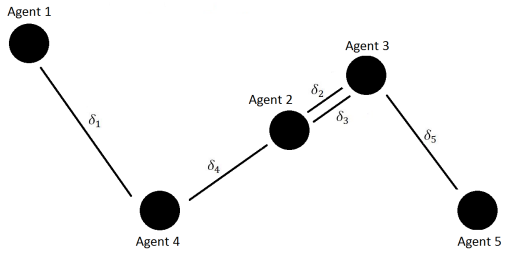


Figure 1: Scatter Domain Representation. Each agent i calculates the distance from itself to the closest agent as δ_i. The global utility is the average of all of these distances. The goal in this domain is for the agents to be as "spread out" as possible.

in the world. At each time step in an episode, an agent takes two actions Δ_x and Δ_y, corresponding to its x and y movements, respectively. The magnitude of these actions are bounded by some upper limit Δ_{max}, which requires that the agents take multiple actions over multiple time steps in order to traverse the domain. The system evaluation function for the scatter domain with N agents is the average minimum distance between agents, given by:

$$G(z(t)) = \frac{\sum_{i=1}^{N} \delta_i(t)}{N} \quad (3)$$

Thus, maximizing the average minimum distance between agents will result in maximizing the system evaluation.

3.2 Rover Domain

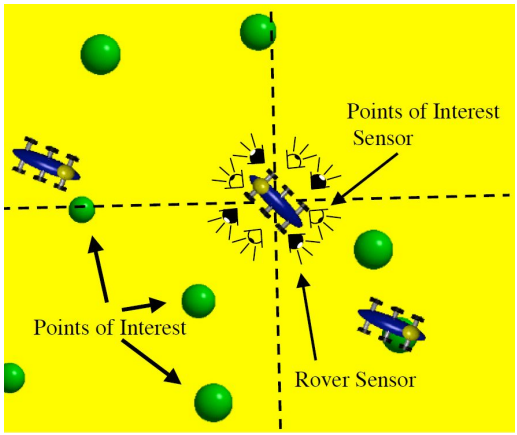


Figure 2: Rover Domain Representation. Each rover senses the closest rover and POI from each of its four sensing quadrants. The rovers must coordinate in order to effectively observe the POIs.

In the rover problem, a collective of rovers on a two dimensional plane aim to observe points of interest (POIs) scattered across the domain (Figure 2). Each POI has an associated value, and each observation of a POI made by a rover yields an observation value which is inversely proportional to the distance that the rover is from the POI. The distance metric used in this domain is the squared Euclidian norm, bounded by a minimum observation value:

$$\delta(x, y) = min\left\{||x - y||^2, \delta^2_{min}\right\} \quad (4)$$

The objective of the rovers is to maximize the observation values of the POIs over the course of an episode, and the system evaluation is calculated as:

$$G = \sum_t \sum_j \frac{V_j}{min_i \delta(L_j, L_{i,t})} \quad (5)$$

where V_j is the value associated with POI j, L_j is the location of POI j, and $L_{i,t}$ is the location of the ith rover at time t.

Although any rover may observe any POI, the system evaluation only takes into account the closest observation made for each POI. In this instantiation of the rover domain, the POI locations are static throughout each experiment. The rovers have eight total sensors, two sensors per quadrant. The rovers sense the closest POI and rover in each of the four quadrants, and these eight readings compose the controller

inputs. The rovers must coordinate in order to achieve high POI coverage. An increasing system evaluation corresponds to better observation coverage of the POIs. As in the scatter domain problem, at each time step the rovers take two actions Δ_x and Δ_y, corresponding to their x and y movements; the magnitude of these actions are bounded by some value Δ_{max}.

4. ALGORITHMS

In this section we provide detailed explanations of the four algorithms investigated in this research, which are:

1. Standard CCEA using system evaluation
2. CCEA using the difference evaluation
3. CCEA using lenient learners and the difference evaluation
4. CCEA using the hall of fame and the difference evaluation

The standard CCEA using the system evaluation is a "standard" CCEA algorithm we use a baseline to assess performance. The second and third algorithms are modifications of existing algorithms which address the problems of suboptimal convergence. The final algorithm is a modification of the hall of fame algorithm combined with the difference evaluation, which also aims to address suboptimal convergence and is the main contribution of this paper.

4.1 Standard CCEA

In the standard CCEA, N coevolving populations of neural networks are utilized to form teams comprised of M agents. In the most general case, M is equal to N. One member of each population is extracted, and these agents are combined to form a team which operate in the problem domain. Each population is initially comprised of k neural networks, randomly initialized. At each generation, k successor networks are generated in each population, which are mutated versions of the parent networks. Then, $2k$ teams of M agents are formed by taking agents from each population and placing these agents into a team. The performance of each of the teams is then evaluated in the domain, and the fitness of every agent in the team is set according to the team's performance. Next, k networks from each population are selected to proceed to the next generation, with the fitness of a network influencing its selection probability. This process is repeated for a set number of generations. The standard CCEA is detailed in Algorithm 1.

In the standard CCEA, each member of a team receives equal credit for that team's performance. This form of credit assignment results in agents' fitness values to be heavily dependent upon the performance of teammates, because each member of the team receives equal credit for the team's performance. The standard CCEA is used as the baseline algorithmm, and serves as a comparison for the other algorithms considered.

```
Initialize N populations of k neural networks
foreach Generation do
    foreach Population do
        produce k successor solutions
        mutate successor solutions
    end
    for i = 1 → 2k do
        randomly select one agent from each population
        add agents to team T_i
        simulate T_i in domain
        assign fitness to each agent in T_i using G(z)
    end
    foreach Population do
        select k networks using ε-greedy
    end
end
```

Algorithm 1: Standard CCEA (See Section 5 for parameters)

4.2 CCEA with the Difference Evaluation

The CCEA with the difference evaluation is carried out in a similar manner to the standard CCEA, except that when a team of agents is evaluated, the fitness of each agent is calculated with the difference evaluation, rather than the system evaluation. Thus, the fitness of each agent of a team is calculated as that agents's contribution to the team's performance, rather than the system evaluation itself. The CCEA with the difference evaluation is equivalent to the CCEA detailed in Algorithm 1, except at the fitness assignment stage, the fitness of each agent is calculated with the difference evaluation rather than the system evaluation.

Thus, the key difference between the CCEA using the difference evaluation and the standard CCEA is credit assignment for the agents. By utilizing the difference evaluation to assign fitness, the fitness of each agent becomes less dependent upon the actions of its teammates. Below, we derive the difference evaluation for the two domains used in this work.

Scatter Domain.

Directly computing the different evaluation using Equation 1 corresponds to simply leaving out agent j from the computation:

$$D_j(z,t) = \frac{\sum_{i \neq j} \delta_i(t)}{N-1} \tag{6}$$

However, this evaluation always increases the system evaluation, because of the nature of the system evaluation function. As the goal in this domain is to maximize average distance between agents, removing any agent will always have a positive effect on the system evaluation. As such agents will need to distinguish between very small positive variations, making the evaluation function in Equation 6 a poor choice for agent fitness function. Instead, in this work, we introduce a *default agent effect*. To compute agent j's fitness then, we replace agent j with this default agent, which yields:

$$D_j = \frac{\sum_{i \neq j} \delta_i(t) + \delta_{def}(t)}{N} \tag{7}$$

where $\delta_{def}(t)$ is a distance associated with the default agent. This distance is set at the beginning of an experiment, and remains constant for each individual calculation of the difference evaluation. This default agent distance corresponds to the c_i (*counterfactual*) term in Equation 1.

Rover Domain.

For the rover problem, the difference evaluation is calcu-

lated by directly applying Equation 1 to Equation 5:

$$D_i(L) = \sum_t \sum_j I_{j,i,t}(z) \left[\frac{V_j}{\delta(L_j, L_{i,t})} - \frac{V_j}{\delta(L_j, L_{k_j,t})} \right] \quad (8)$$

where k_j is the second closest rover to POI j, and $I_{j,i,t}(z)$ is an indicator function which returns 1.0 if and only if rover i is the closest rover to POI j at time t. If rover i is not the closest rover to any POI at time t, then its difference evaluation is zero, indicating that the rover is not contributing to the system utility at time t.

4.3 CCEA with Lenient Learners and Difference Evaluation

The CCEA with lenient learners and the difference evaluation is carried out in a similar manner to the CCEA with the difference evaluation, except that each agent is tested with multiple sets of collaborators (i.e. the agent will be placed in multiple teams), and the highest fitness achieved is the fitness assigned to that agent. As the algorithm progresses, agents become less lenient learners; that is, they are tested against fewer sets of collaborators. The CCEA with lenient learners and the difference evaluation is detailed in Algorithm 2.

```
Initialize N populations of k neural networks
foreach Generation do
    foreach Population do
        produce k successor solutions
        mutate successor solutions
    end
    for i = 1 → 2k · m do
        randomly select one agent from each population
        add agents to team T_i
        simulate T_i in domain
        assign fitness to each agent in T_i using D(z)
    end
    foreach Population do
        foreach Member do
            fitness ← maximum fitness attained
        end
    end
    foreach Population do
        select k networks using ε-greedy
    end
end
```

Algorithm 2: Lenient CCEA using Difference Evaluation (See Section 5 for parameters)

It is important to note that in Algorithm 2, each member of each population is selected exactly m times, and the value of m is decreased as the algorithm progresses. This corresponds to agents being lenient in the early stages of evolution, and becoming less and less lenient as evolutionary time passes. By incorporating leniency and the difference evaluation into the CCEA, the effects of an agent's teammates on the fitness of that agent are minimized.

4.4 CCEA with Hall of Fame and Difference Evaluation

As noted in Section 2, CCEAs have been shown to provide better solutions when the fitness of an individual is based partly on how it performs with its team, and partly on how it would perform if it were paired with optimal collaborators. However, estimating the fitness of an agent paired with optimal collaborators is a difficult task, especially in complex domains. Rather than estimating what optimal collaborators would be for a particular agent, the *hall of fame* method is altered to estimate the behavior of optimal collaborators. The CCEA with the hall of fame and difference evaluation is carried out in a similar manner to the CCEA with the difference evaluation, except that the fitness assignment is altered. At the end of each generation, the team that achieves the highest system evaluation is compared against the hall of fame members. If that team achieved a higher system evaluation than all of the hall of fame members, then that team is added to the hall of fame. When assigning fitness to each agent of a team, the difference evaluation of that agent is calculated, as well as the difference utility of that agent when it replaces an agent from the best hall of fame team. The performance of the best hall of fame team is non-decreasing with respect to evolutionary time, so this team approaches the optimal team as the CCEA progresses. The difference evaluation of an agent when compared with the hall of fame team is calculated as:

$$D_{HOF,i} = G_{HOF+i} - G_{HOF} \quad (9)$$

where G_{HOF+i} is the system evaluation of the best hall of fame team when agent i replaces the corresponding member of the hall of fame team, and G_{HOF} is the system evaluation of the best hall of fame team. So, in the CCEA with the hall of fame and difference evaluation, the fitness of an agent is calculated as:

$$F(i) = \alpha \cdot D_i + (1 - \alpha) \cdot D_{HOF,i} \quad (10)$$

where D_i is the difference evaluation of agent i when collaborating with its team, $D_{HOF,i}$ is the agent's difference evaluation when paired with the best hall of fame team as in Equation 9, and $\alpha \in [0, 1]$ is a weight corresponding to the relative importance of the difference evaluation and the difference evaluation with estimated optimal collaborators. The CCEA with the hall of fame and difference evaluation is detailed in Algorithm 3.

By assuming that the best hall of fame team is the set of optimal collaborators for any agent, the complexities of estimating what a set of optimal collaborators would be are eliminated. The CCEA using the difference evaluation and the hall of fame includes shaped fitness functions to tell agents what their individual contributions to team performance are, as well as biasing the fitness functions using the concept of estimated optimal collaborators via the hall of fame. This approach modifies the hall of fame algorithm in two ways. First, the hall of fame is now comprised of *teams*, rather than *individuals*. Secondly, the hall of fame is now utilized in *cooperative* coevolution, rather than *competitive* coevolution.

5. EXPERIMENTAL RESULTS

The algorithms outlined in section 4 were all tested in the scatter domain and the rover domain, with team sizes varying from 10 to 100 agents. For experiments with 10 agent teams, 10 coevolving populations of 200 members each were used. For experiments with 100 agent teams, 100 coevolving populations of 25 members each were utilized. For the standard CCEA, Equation 3 with a default agent distance

```
Initialize N populations of k neural networks
foreach Generation do
    foreach Population do
        produce k successor solutions
        mutate successor solutions
    end
    for i = 1 → 2k · m do
        randomly select one agent from each population
        add agents to team T_i
        simulate T_i in domain
        assign fitness to each agent with Eq. 10
    end
    foreach Team T_i do
        if G(z|T_i) > G(z|HOF_best) then
            add T_i to HOF
        end
    end
    foreach Population do
        select k networks using ϵ-greedy
    end
end
```

Algorithm 3: CCEA using Difference Evaluation and Hall of Fame (See Section 5 for parameters)

of 1.0 was used to assign fitness in the scatter domain and Equation 5 was used in the rover domain. For the CCEA algorithm with the difference evaluation, Equation 7 was used to assign fitness for the scatter domain and Equation 8 for the rover domain. For the CCEA with lenient learners and the difference evaluation, the same fitness equations were used as in the CCEA with the difference evaluation experiments. The leniency value m was initially set to 10, and decreased by 1 every 200 generations, resulting in no leniency after 2000 generations. The CCEA with the hall of fame and difference evaluation had the same fitness assignments as in the CCEA with the difference utility. When calculating the fitness from equation 9, α was set to 0.5. For all experiments, network mutation was carried out by adding values drawn from a Gaussian distribution to network weights. In the beginning of the evolution, one weight per network was mutated with a standard deviation of 1.0. At the end of the evolution, each network weight was mutated with a standard deviation of 0.1. These mutation parameters were varied linearly throughout the evolutionary process. For each experiment, 100 statistical runs were completed, with the standard error in the mean ($\sigma/\sqrt{N}$) being reported. The experiment details and results are given in the following sections.

5.1 Scatter Domain

First, we applied each of the four CCEA algorithms to the scatter domain. For the 10 agent experiment, the world was set to a 10 by 10 plane world and run for 10 time steps, and Δ_{max} was set to 1.0. For the 100 agent experiment, the world was set to a 31.6 by 31.6 plane world and run for 10 time steps, and Δ_{max} was set to 3.16. These values were chosen such that the plane area to number of agents ratio was constant for each experiment, and the agents could traverse the world in the same number of time steps. At the beginning of each experiment, the agents all started at the center of the plane worlds. Figure 3 shows the learning curve for the 10 agent problem. Figure 4 shows the learning curve for the 100 agent problem. Finally, Figure 5 shows the

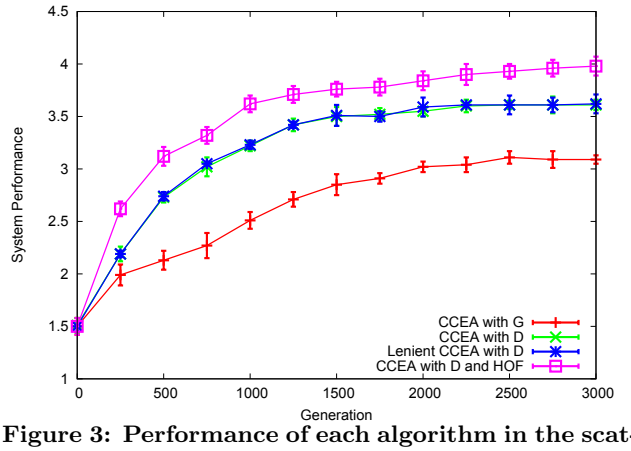


Figure 3: Performance of each algorithm in the scatter domain, with 10 agents. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods tested.

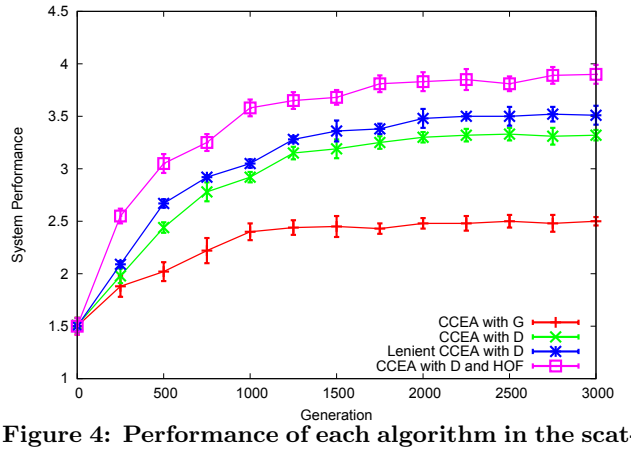


Figure 4: Performance of each algorithm in the scatter domain, with 100 agents. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods tested.

scaling properties of each algorithm in the scatter domain.

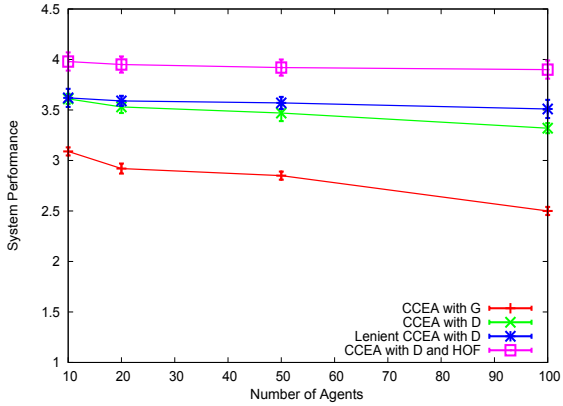


Figure 5: Scaling Performance in Scatter Domain. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods tested, and this difference in performance increases with team size. The addition of leniency does not significantly help when there are a small number of agents, but as the team size goes up, leniency becomes increasingly useful.

All three algorithms using the different evaluation function outperformed the standard CCEA in this domain. This is not surprising, because credit assignment in this algorithm is highly subjective, and the fitness of each agent was greatly influenced by its teammates. The CCEA with the difference evaluation and the CCEA with leniency and the difference utility performed almost identically in the 10 agent case, but the addition of leniency provided improved performance in the 100 agent case. This is an interesting result, and gives insight to the properties of leniency. Leniency and the difference utility both perform similar functions. Leniency aims to reduce the subjectiveness of credit assignments in CCEAs by partnering agents with multiple sets of collaborators. By testing an agent with multiple teams and taking the highest fitness achieved, the likelihood that an agent's fitness is too strongly biased by its teammates is minimized. The difference utility also reduces the subjectiveness of credit assignment, by isolating an agent's individual contribution to its team's performance. Thus, leniency and the difference evaluation achieve a similar goal, although in different manners. However, as the problem becomes more complex by increasing the team size, coupling both approaches provided benefits, as seen in Figure 5. As the team size is increased, the performance of the CCEA with the difference evaluation and lenient learners outperforms the CCEA with the difference reward by larger and larger margins.

The CCEA with the hall of fame and difference evaluation performed the best out of all of the algorithms tested. The goal of introducing the hall of fame was to approximate optimal collaborators, in order to bias the CCEA toward optimal solutions. The scatter domain results show that approximating optimal collaborators with the best known team is an acceptable approach to bias the CCEA. The hall of fame approach has the benefit that it is much simpler to implement than an ad hoc estimate of optimal collaborators, and is a more attractive alternative as the domain becomes more complex, because such an ad hoc estimate becomes increasingly difficult to develop as the domain complexifies. As seen in Figure 5, this approach becomes better compared to the other approaches as the number of agents increases.

5.2 Rover Domain

Finally, we applied each of the four CCEA algorithms to the rover domain. At the beginning of each experiment, n POIs were placed randomly in the domain, and their positions remained constant throughout each experiment, where n is equivalent to the number of agents in the domain. The minimum observation distance δ_{min} was set to 0.1. The simulations were carried out in a 10 by 10 world for 25 time steps, and Δ_{max} was set to 1.0. At the beginning of each simulation, each rover started in the center of the world. Each algorithm was tested over 50 statistical runs. Figure 6 shows the results for 10 agent teams, Figure 7 shows the results for 100 agent teams, and Figure 8 shows the scaling results for the rover domain.

As in the scatter domain experiments, all three algorithms using the different evaluation function outperformed the standard CCEA in the rover domain. This can be attributed to the fact that the system evaluation does not give an agent good feedback on its individual contribution to the team's performance, and the agent thus has a difficult time learning an optimal policy. In the scatter domain, the CCEA with the difference evaluation and the CCEA with lenient learners and the difference utility performed nearly identically with 10 agent teams, but leniency became more important as the team size went up. In the more complex rover domain, the CCEA with lenient learners and the difference evaluation performed slightly better than the CCEA with the difference reward, and this difference also increased with the team size. This indicates that although leniency and the difference utility have similar effects on the learning process, leniency may become more beneficial in CCEAs as the domain becomes more complex or the number of cooperating agents increases.

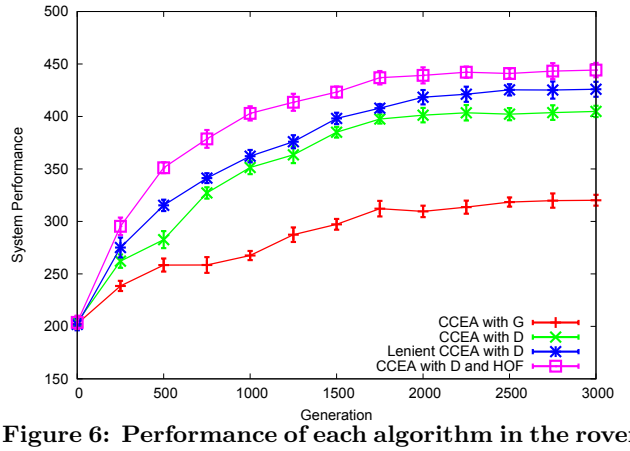


Figure 6: Performance of each algorithm in the rover domain, with 10 agents and 10 POIs. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods tested.

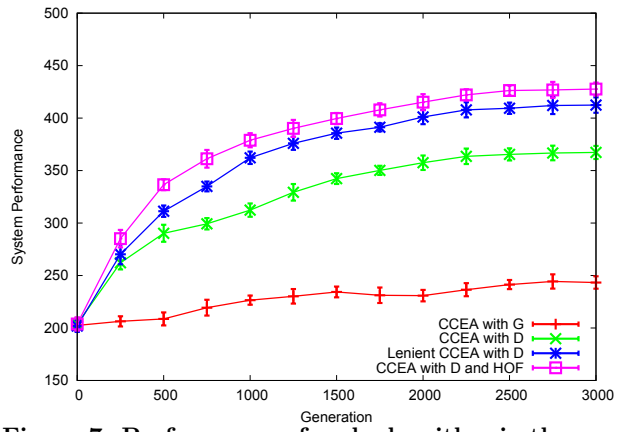


Figure 7: Performance of each algorithm in the rover domain, with 100 agents. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods tested.

As in the scatter domain, the CCEA with the hall of fame and difference evaluation performed the best out of all algorithms tested. This further supports the conclusion that biasing the CCEA with hall of fame teams approximating optimal collaborators, as well as shaping the fitness functions, helps guide the search towards better solutions. As seen in Figure 8, the difference in performance between the CCEA with the difference evaluation and hall of fame and the CCEA with the global reward increases with team size, indicating that for increasingly complex systems, this new algorithm becomes more and more useful. In all the experiments that were conducted, the CCEA with the hall of fame and difference evaluation significantly outperformed

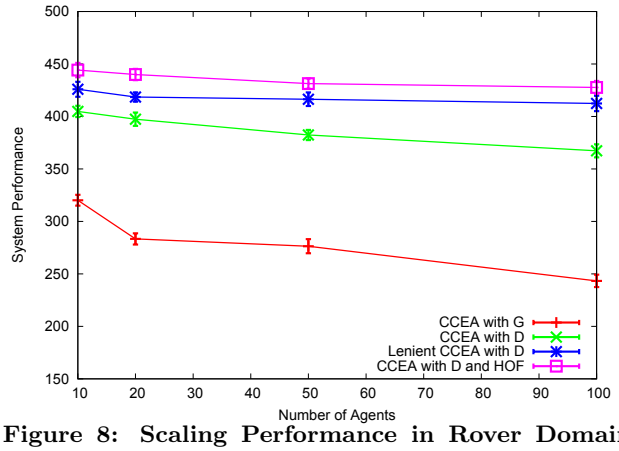


Figure 8: Scaling Performance in Rover Domain. The CCEA with the difference evaluation and hall of fame biasing outperforms all other methods, and the gap in performance increases with team size.

all other algorithms, and was able to easily estimate optimal collaborators in order to bias the fitness. This fitness biasing, in addition to shaping the fitness values with the difference evaluation, contributed to significantly better performance than any other algorithm tested.

6. DISCUSSION

This paper presented three CCEA algorithms where leniency and hall of fame methods were used in combination with fitness shaping. Combining hall of fame and difference evaluation outperformed all other algorithms in two different domains. It is known that shaping fitness functions can greatly improve the efficacy of a CCEA [8], but this often isn't enough to obtain optimal performance. Biasing a CCEA with an estimate optimal collaborators results in a more effective search for optimal policies, but this estimate is problematic because it is an ad hoc, domain dependent estimate [10]. Our algorithm circumvents the problem of estimating optimal collaborators, so the CCEA search is easily biased in a domain independent fashion. Furthermore, the algorithm shapes agent fitnesses in order to provide each agent a measure of its individual contribution to the system objective.

A particularly interesting result was that in the simpler scatter domain, using the difference evaluation to shape the fitness functions performed equivalently to a method which used lenient learners and the difference utility for 10 agent teams. Intuitively, the difference utility and lenient learners both aim to achieve the same goal, which is to isolate an agent's individual contribution to the system evaluation. This is one key reason their performance was similar in the simpler domains. However, with larger teams or a more complicated domain, leniency in addition to the difference evaluation performed better than the difference evaluation alone. Our current research focuses in two directions: (i) investigating the theoretical relationship between leniency and difference evaluation functions, including determining when adding leniency to fitness shaping is desirable; and (ii) the impact of biasing the search while using difference evaluation functions, including alternative biasing methods.

Acknowledgements

This work was partially supported by NSF Grant IIS-0910358, and DoE NETL grant DE-FE0000857.

7. REFERENCES

[1] A. K. Agogino and K. Tumer. Efficient evaluation functions for evolving coordination. *Evolutionary Computation*, 16(2):257–288, 2008.

[2] A. K. Agogino and K. Tumer. A multiagent approach to managing air traffic flow. *Journal of Autonomous Agents and Multi-Agent Systems*, 24:1–25, 2012. DOI: 10.1007/s10458-010-9142-5.

[3] A. Bucci and J. B. Pollack. Thoughts on solution concepts. In *"Proceedings of the Genetic and Evolutionary Computation Conference – GECCO-2007*. ACM, 2007.

[4] S. G. Ficici. Monotonic solution concepts in coevolution, 2005.

[5] S. G. Ficici, O. Melnik, and J. Pollack. A game-theoretic and dynamical-systems analysis of selection methods in coevolution, 2005.

[6] D. Fogel. An introduction to simulated evolutionary optimization. *Neural Networks, IEEE Transactions on*, 5(1):3 –14, jan 1994.

[7] T. D. Haynes, D. A. Schoenefeld, and R. L. Wainwright. Type inheritance in strongly typed genetic programming. In *Advances in Genetic Programming 2, chapter 18*, pages 359–376. MIT Press, 1996.

[8] P. J. Hoen and E. D. D. Jong. Evolutionary multi-agent systems. In *In Proceedings of the 8th International Conference on Parallel Problem Solving from Nature PPSN-04*, pages 872–881, 2004.

[9] M. Knudson and K. Tumer. Coevolution of heterogeneous multi-robot teams. In *Proceedings of the Genetic and Evolutionary Computation Conference*, Portland, OR, July 2010.

[10] L. Panait, S. Luke, and R. P. Wiegand. Biasing coevolutionary search for optimal multiagent behaviors. *IEEE Transactions on Evolutionary Computation*, 10(6):629–645, 2006.

[11] L. Panait, K. Tuyls, and S. Luke. Theoretical advantages of lenient learners: An evolutionary game theoretic perspective. *J. Mach. Learn. Res.*, 9:423–457, June 2008.

[12] M. A. Potter and K. A. De Jong. Evolving Neural Networks with Collaborative Species. *Computer Simulation Conference*, 1995.

[13] C. Rosin and R. Belew. New methods for competitive coevolution. *Evolutionary Computation*, 5:1–29, 1996.

[14] T. Soule and R. B. Heckendorn. A developmental approach to evolving scalable hierarchies for multi-agent swarms. In *"Proceedings of the 12th Annual Conference on Genetic and Evolutionary Compuation – GECCO-2010*. ACM, 2010.

[15] P. Stone and M. Veloso. Multiagent systems: A survey from a machine learning perspective. *AUTONOMOUS ROBOTS*, 8:345–383, 1997.

[16] R. P. Wiegand, K. A. D. Jong, and W. C. Liles. Modeling variation in cooperative coevolution using evolutionary game theory, 2002.

Dynamic Potential-Based Reward Shaping

Sam Devlin
Department of Computer Science,
University of York, UK
devlin@cs.york.ac.uk

Daniel Kudenko
Department of Computer Science,
University of York, UK
kudenko@cs.york.ac.uk

ABSTRACT

Potential-based reward shaping can significantly improve the time needed to learn an optimal policy and, in multi-agent systems, the performance of the final joint-policy. It has been proven to not alter the optimal policy of an agent learning alone or the Nash equilibria of multiple agents learning together.

However, a limitation of existing proofs is the assumption that the potential of a state does not change dynamically during the learning. This assumption often is broken, especially if the reward-shaping function is generated automatically.

In this paper we prove and demonstrate a method of extending potential-based reward shaping to allow dynamic shaping and maintain the guarantees of policy invariance in the single-agent case and consistent Nash equilibria in the multi-agent case.

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

1. INTRODUCTION

Reinforcement learning agents are typically implemented with no prior knowledge and yet it has been repeatedly shown that informing the agents of heuristic knowledge can be beneficial [2, 7, 13, 14, 17, 19]. Such prior knowledge can be encoded into the initial Q-values of an agent or the reward function. If done so by a potential function, the two can be equivalent [23].

Originally potential-based reward shaping was proven to not change the optimal policy of a single agent provided a static potential function based on states alone [15]. Continuing interest in this method has expanded its capabilities to providing similar guarantees when potentials are based on states and actions [24] or the agent is not alone but acting in a common environment with other shaped or unshaped agents [8].

However, all existing proofs presume a static potential function. A static potential function represents static knowledge and, therefore, can not be updated online whilst an agent is learning.

Despite these limitations in the theoretical results, empirical work has demonstrated the usefulness of a dynamic potential function [10, 11, 12, 13]. When applying potential-based reward shaping, a common challenge is how to set the potential function. The existing implementations using dynamic potential functions automate this process making the method more accessible to all.

Some, but not all, pre-existing implementations enforce that their potential function stabilises before the agent. This feature is perhaps based on the intuitive argument that an agent cannot converge until the reward function does so [12]. However, as we will show in this paper, agents can converge despite additional dynamic rewards provided they are of a given form.

Our contribution is to prove how a dynamic potential function does not alter the optimal policy of a single-agent problem domain or the Nash equilibria of a multi-agent system (MAS). This proof justifies the existing uses of dynamic potential functions and explains how, in the case where the additional rewards are never guaranteed to converge [10], the agent can still converge.

Furthermore, we will also prove that, by allowing the potential of state to change over time, dynamic potential-based reward shaping is not equivalent to Q-table initialisation. Instead it is a unique tool, useful for developers wishing to continually influence an agent's exploration whilst guaranteed to not alter the goal(s) of an agent or group.

In the next section we will cover all relevant background material. In Section 3 we present both of our proofs regarding the implications of a dynamic potential function on existing results in potential-based reward shaping. Later, in Section 4, we clarify our point by empirically demonstrating a dynamic potential function in both single-agent and multi-agent problem domains. The paper then closes by summarising the key results of the paper.

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

2. PRELIMINARIES

In this section we introduce all relevant existing work upon which this work is based.

2.1 Reinforcement Learning

Reinforcement learning is a paradigm which allows agents to learn by reward and punishment from interactions with the environment [21]. 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 [16].

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 reward function) are available, this task can be solved using policy iteration [3].

When the environment dynamics are not available, as with most real problem domains, policy 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 pairs, $Q(s, a)$ [20]. 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 [21]. 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'.

Provided each state-action pair is experienced an infinite number of times, the rewards are bounded and the agent's exploration and learning rate reduce to zero the value table of a Q-learning agent will converge to the optimal values Q^* [22].

2.1.1 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 latter is a group of multi-agent specific algorithms designed to consider the existence of other agents. The former is the deployment of multiple agents each using a single-agent reinforcement learning algorithm.

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 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. 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.

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 [18].

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 $\langle S, A_1, ..., A_n, T, R_1, ..., R_n\rangle$, where S is the state space, A_i is the action space of agent i, $T(s, a_{i...n}, s') = Pr(s'|s, a_{i...n})$ is the probability that joint action $a_{i...n}$ in state s will lead to state s', and $R_i(s, a_i, s')$ is the immediate reward received by agent i when taking action a_i in state s results in a transition to state s' [9].

Typically, reinforcement learning agents, whether alone or sharing an environment, are deployed with no prior knowledge. The assumption is that the developer has no knowledge of how the agent(s) should behave. However, more often than not, this is not the case and the agent(s) can benefit from the developer's understanding of the problem domain.

One common method of imparting knowledge to a reinforcement learning agent is reward shaping, a topic we will discuss in more detail in the next subsection.

2.2 Reward Shaping

The idea of reward shaping is to provide an additional reward representative of prior knowledge to reduce the number of suboptimal actions made and so reduce the time needed to learn [15, 17]. 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 any state-based 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 [17]. To deal with such problems, potential-based reward shaping was proposed [15] 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. [15] 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 [23] later proved that an agent learning with potential-based reward shaping and no knowledge-based Q-table initialisation will behave identically to an agent with-

out reward shaping when the latter agent's value function is initialised with the same potential function.

These proofs, and all subsequent proofs regarding potential-based reward shaping including those presented in this paper, require actions to be selected by an advantage-based policy [23]. Advantage-based policies select actions based on their relative differences in value and not their exact value. Common examples include greedy, ϵ-greedy and Boltzmann soft-max.

2.2.1 Reward Shaping In Multi-Agent Systems

Incorporating heuristic knowledge has been shown to also be beneficial in multi-agent reinforcement learning [2, 13, 14, 19]. However, some of these examples did not use potential-based functions to shape the reward [14, 19] and could, therefore, potentially suffer from introducing beneficial cyclic policies that cause convergence to an unintended behaviour as demonstrated previously in a single-agent problem domain [17].

The remaining applications that were potential-based [2, 13], demonstrated an increased probability of convergence to a higher value Nash equilibrium. However, both of these applications were published with no consideration of whether the proofs of guaranteed policy invariance hold in multi-agent reinforcement learning.

Since this time, theoretical results [8] have shown that whilst Wiewiora's proof [23] of equivalence to Q-table initialisation holds also for multi-agent reinforcement learning Ng's proof [15] of policy invariance does not. Multi-agent potential-based reward shaping can alter the final policy a group of agents will learn but, instead, does not alter the Nash equilibria of the system.

2.2.2 Dynamic Reward Shaping

Reward shaping is typically implemented bespoke for each new environment using domain-specific heuristic knowledge [2, 7, 17] but some attempts have been made to automate [10, 11, 12, 13] the encoding of knowledge into a potential function.

All of these existing methods alter the potential of states online whilst the agent is learning. Neither the existing single-agent [15] nor the multi-agent [8] proven theoretical results considered such dynamic shaping.

However, the opinion has been published that the potential function must converge before the agent can [12]. In the majority of implementations this approach has been applied [11, 12, 13] but in other implementations stability is never guaranteed [10]. In this case, despite common intuition, the agent was still seen to converge to an optimal policy.

Therefore, contrary to existing opinion it must be possible for an agent's policy to converge despite a continually changing reward transformation. In the next section we will prove how this is possible.

3. THEORY

In this section we will cover the implications of a dynamic potential function on the three most important existing proofs in potential-based reward shaping. Specifically, in subsection 3.1 we address the theoretical guarantees of policy invariance in single-agent problem domains [15] and consistent Nash equilibria in multi-agent problem domains [8]. Later, in subsection 3.2, we will address Wiewiora's proof of equivalence to Q-table initialisation [23].

3.1 Dynamic Potential-Based Reward Shaping Can Maintain Existing Guarantees

To extend potential-based reward shaping to allow for a dynamic potential function we extend Equation 3 to include time as a parameter of the potential function Φ. Informally, if the difference in potential is calculated from the potentials of the states at the time they were visited the guarantees of policy invariance or consistant Nash equilibria remain. Formally:

$$F(s,t,s',t') = \gamma\Phi(s',t') - \Phi(s,t) \tag{4}$$

where t is the time the agent arrived at previous state s and t' is the current time when arriving at the current state s' (i.e. $t < t'$).

To prove policy invariance in the single-agent case and consistent Nash equilibria in the multi-agent case it suffices to show that the return a shaped agent will receive for following a fixed sequence of states and actions is equal to the return the non-shaped agent would receive when following the same sequence minus the potential of the first state in the sequence [1, 8].

Therefore, let us consider the return U_i for any arbitrary agent i when experiencing sequence $\bar{s}$ in a discounted framework without shaping. Formally:

$$U_i(\bar{s}) = \sum_{j=0}^{\infty} \gamma^j r_{j,i} \tag{5}$$

where $r_{j,i}$ is the reward received at time j by agent i from the environment.

Given this definition of return, the true Q-values can be defined formally by:

$$Q_i^*(s,a) = \sum_{\bar{s}} Pr(\bar{s}|s,a)U_i(\bar{s}) \tag{6}$$

Now consider the same agent but with a reward function modified by adding a dynamic potential-based reward function of the form given in Equation 4. The return of the shaped agent $U_{i,\Phi}$ experiencing the same sequence $\bar{s}$ is:

$$\begin{aligned}
U_{i,\Phi}(\bar{s}) &= \sum_{j=0}^{\infty} \gamma^j (r_{j,i} + F(s_j,t_j,s_{j+1},t_{j+1})) \\
&= \sum_{j=0}^{\infty} \gamma^j (r_{j,i} + \gamma\Phi(s_{j+1},t_{j+1}) - \Phi(s_j,t_j)) \\
&= \sum_{j=0}^{\infty} \gamma^j r_{j,i} + \sum_{j=0}^{\infty} \gamma^{j+1}\Phi(s_{j+1},t_{j+1}) \\
&\quad - \sum_{j=0}^{\infty} \gamma^j \Phi(s_j,t_j) \\
&= U_i(\bar{s}) + \sum_{j=1}^{\infty} \gamma^j \Phi(s_j,t_j) \\
&\quad - \sum_{j=1}^{\infty} \gamma^j \Phi(s_j,t_j) - \Phi(s_0,t_0) \\
&= U_i(\bar{s}) - \Phi(s_0,t_0)
\end{aligned} \tag{7}$$

Then by combining 6 and 7 we know the shaped Q-function is:

$$
\begin{aligned}
Q^*_{i,\Phi}(s,a) &= \sum_{\bar{s}} Pr(\bar{s}|s,a)U_{i,\Phi}(\bar{s}) \\
&= \sum_{\bar{s}} Pr(\bar{s}|s,a)(U_i(\bar{s}) - \Phi(s,t)) \\
&= \sum_{\bar{s}} Pr(\bar{s}|s,a)U_i(\bar{s}) - \sum_{\bar{s}} Pr(\bar{s}|s,a)\Phi(s,t) \\
&= Q^*_i(s,a) - \Phi(s,t) \qquad (8)
\end{aligned}
$$

where t is the current time.

As the difference between the original Q-values and the shaped Q-values is not dependent on the action taken, then in any given state the best (or best response) action remains constant regardless of shaping. Therefore, we can conclude that the guarantees of policy invariance and consistent Nash equilibria remain.

3.2 Dynamic Potential-Based Reward Shaping Is Not Equivalent To Q-Table Initialisation

In both single-agent [23] and multi-agent [8] reinforcement learning, potential-based reward shaping with a static potential function is equivalent to initialising the agent's Q-table such that:

$$\forall s, a | Q(s,a) = \Phi(s) \qquad (9)$$

where $\Phi(\cdot)$ is the same potential function as used by the shaped agent.

However, with a dynamic potential function this result no longer holds. The proofs require an agent with potential-based reward shaping and an agent with the above Q-table initialisation to have an identical probability distribution over their next action provided the same history of states, actions and rewards.

If the Q-table is initialised with the potential of states prior to experiments ($\Phi(s,t_0)$), then any future changes in potential are not accounted for in the initialised agent. Therefore, after the agents experience a state where the shaped agent's potential function has changed they may make different subsequent action choices.

Formally this can be proved by considering agent L that receives dynamic potential-based reward shaping and agent L' that does not but is initialised as in Equation 9. Agent L will update its Q-values by the rule:

$$
\begin{aligned}
Q(s,a) \leftarrow\ & Q(s,a) + \\
& \underbrace{\alpha\left(r_i + F(s,t,s',t') + \gamma \max_{a'} Q(s',a') - Q(s,a)\right)}_{\delta Q(s,a)}
\end{aligned} \qquad (10)
$$

where $\Delta Q(s,a) = \alpha \delta Q(s,a)$ is the amount 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:

$$Q(s,a) = Q_0(s,a) + \Delta Q(s,a) \qquad (11)$$

where $Q_0(s,a)$ is the initial Q-value of state-action pair (s,a). Similarly, agent L' updates its Q-values by the rule:

$$
Q'(s,a) \leftarrow Q'(s,a) + \underbrace{\alpha\left(r_i + \gamma \max_{a'} Q'(s',a') - Q'(s,a)\right)}_{\delta Q'(s,a)} \qquad (12)
$$

And its current Q-values can be represented formally as:

$$Q'(s,a) = Q_0(s,a) + \Phi(s,t_0) + \Delta Q'(s,a) \qquad (13)$$

where $\Phi(s,t_0)$ is the potential for state s before learning begins.

For the two agents to act the same they must choose their actions by relative difference in Q-values, not absolute magnitude, and the relative ordering of actions must remain the same for both agents. Formally:

$$\forall s, a, a' | Q(s,a) > Q(s,a') \Leftrightarrow Q'(s,a) > Q'(s,a') \qquad (14)$$

In the base case this remains true, as both $\Delta Q(s,a)$ and $\Delta Q'(s,a)$ equal zero before any actions are taken, but after this the proof falters for dynamic potential functions.

Specifically, when the agents first transition to a state where the potential has changed agent L will update $Q(s,a)$ by:

$$
\begin{aligned}
\delta Q(s,a) &= r_i + F(s,s') + \gamma \max_{a'} Q(s',a') - Q(s,a) \\
&= r_i + \gamma\Phi(s',t') - \Phi(s,t) \\
&\quad + \gamma \max_{a'}(Q_0(s',a') + \Delta Q(s',a')) \\
&\quad - Q_0(s,a) - \Delta Q(s,a) \\
&= r_i + \gamma\Phi(s',t') - \Phi(s,t_0) \\
&\quad + \gamma \max_{a'}(Q_0(s',a') + \Delta Q(s',a')) \\
&\quad - Q_0(s,a) - \Delta Q(s,a) \qquad (15)
\end{aligned}
$$

and agent L' will update $Q'(s,a)$ by:

$$
\begin{aligned}
\delta Q'(s,a) &= r_i + \gamma \max_{a'} Q'(s',a') - Q'(s,a) \\
&= r_i + \gamma \max_{a'}(Q_0(s',a') + \Phi(s',t_0) + \Delta Q'(s'a')) \\
&\quad - Q_0(s,a) - \Phi(s,t_0) - \Delta Q'(s,a) \\
&= r_i + \gamma \max_{a'}(Q_0(s',a') + \Phi(s',t_0) + \Delta Q(s'a')) \\
&\quad - Q_0(s,a) - \Phi(s,t_0) - \Delta Q(s,a) \\
&= r_i + \gamma\Phi(s',t_0) - \Phi(s,t_0) \\
&\quad + \gamma \max_{a'}(Q_0(s',a') + \Delta Q(s',a')) \\
&\quad - Q_0(s,a) - \Delta Q(s,a) \\
&= \delta Q(s,a) - \gamma\Phi(s',t') + \gamma\Phi(s',t_0) \qquad (16)
\end{aligned}
$$

But the two are not equal as:

$$\Phi(s',t') \neq \Phi(s',t_0) \qquad (17)$$

Therefore, for this state-action pair:

$$Q'(s,a) = Q(s,a) + \Phi(s,t_0) - \alpha\gamma\Phi(s',t') + \alpha\gamma\Phi(s',t_0) \qquad (18)$$

but for all other actions in state s:

$$Q'(s,a) = Q(s,a) + \Phi(s,t_0) \qquad (19)$$

Once this occurs the differences in Q-values between agent L and agent L' for state s would no longer be constant across all actions. If this difference is sufficient to change the ordering of actions (i.e. Equation 14 is broken), then the policy of any rational agent will have different probability distributions over subsequent action choices in state s.

In single-agent problem domains, provided the standard necessary conditions are met, the difference in ordering will only be temporary as agents initialised with a static-potential function and/or those receiving dynamic potential-based reward shaping will converge to the optimal policy. In these cases the temporary difference will only affect the exploration of the agents not their goal.

In multi-agent cases, as was shown previously [8], altered exploration can alter final joint-policy and, therefore, the different ordering may remain. However, as we have proven in the previous sub-section, this is not indicative of a change in the goals of the agents.

In both cases, we have shown how an agent initialised as in Equation 9 can after the same experiences behave differently to an agent receiving dynamic potential-based reward shaping. This occurs because the initial value given to a state cannot capture subsequent changes in it's potential.

Alternatively, the initialised agent could reset its Q-table on each change in potential to reflect the changes in the shaped agent. However, this approach would lose all history of updates due to experiences had and so again cause differences in behaviour between the shaped agent and the initialised agent.

Furthermore, this method and other similar methods of attempting to integrate change in potential after the agent has begun to learn are also no longer strictly Q-table initialisation.

Therefore, we conclude that there is not a method of initialising an agent's Q-table to guarantee equivalent behaviour to an agent receiving dynamic potential-based reward shaping.

4. EMPIRICAL DEMONSTRATION

To clarify our contribution in the following subsections we will demonstrate empirically for both a single-agent and a multi-agent problem domain that their respective guarantees remain despite a dynamic potential function. Specifically in both environments we implement agents without shaping or with a (uniform or negatively biased) random potential function that never stabilises.

4.1 Single-Agent Example

To demonstrate policy invariance with and without dynamic potential-based reward shaping, an empirical study of a discrete, deterministic grid world will be presented here.

Specifically we have one agent attempting to move from grid location S to G in the maze illustrated in Figure 1. The optimal policy/route through the maze takes 41 time steps and should be learnt by the agent regardless of whether it does or does not receive the reward shaping.

On each time step the agent receives -1 reward from the environment. Upon reaching the goal the agent receives $+100$ reward from the environment. If an episode reaches 1000 time steps without reaching the goal, the episode is reset.

At each time step, if the agent is receiving uniform random shaping, the state entered will be given a random potential between 0 and 50 and the agent will receive an additional reward equal to the difference between this new potential[1] and the potential of the previous state.

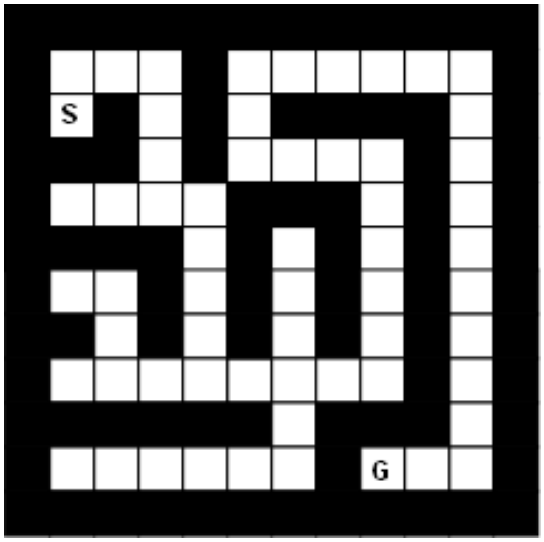


Figure 1: Map of Maze

Likewise, if the agent is receiving negative bias random shaping, the state entered will be given a random potential between 0 and it's current distance to the goal. This potential function is dynamic, never stabilises and encourages movement away from the agent's goal.

The agent implemented uses Q-learning with ϵ-greedy exploration and a tabular representation of the environment. Experimental parameters were set as $\alpha = 0.05$,$\gamma = 1.0$ and ϵ begins at 0.4 and reduces linearly over the first 500 episodes to 0.

4.1.1 Results

All experiments were run for 1000 episodes and repeated 100 times. The results, illustrated in Figure 2, plot the mean number of steps taken to complete that episode. All figures include error bars illustrating the standard error from the mean.

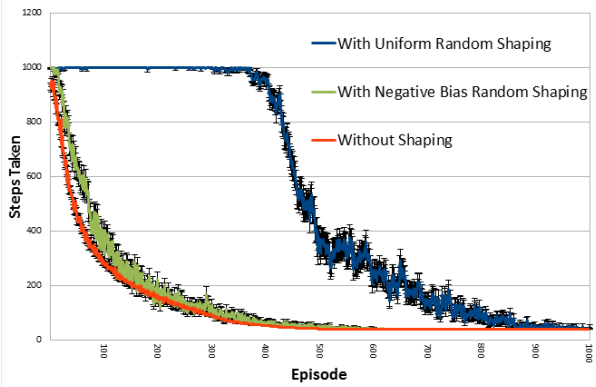


Figure 2: Single-Agent Maze Results

[1]If γ was less than 1 then this value would be discounted by γ, as we will demonstrate in the multi-agent example.

As we expected, regardless of shaping, the agent learns the optimal policy and can complete the maze within 41 time steps. This is the first published example of a reinforcement learning agent converging despite a reward shaping function that is known not to converge. This example counters the previously accepted intuition [12] and supports our claim that the guarantee of policy invariance remains provided the additional reward is of the form:

$$F(s, s') = \gamma\Phi(s', t') - \Phi(s, t)$$

In this example, the agents with dynamic potential-based reward shaping take longer to learn the optimal policy. However, this is not characteristic of the method but of our specific potential functions. For this problem domain, a uniform random potential-function, has been shown to be the worst possible case. This is because it represents no specific knowledge whilst the negative bias random potential function encourages movement away from the goal which in some parts of the maze is the correct behaviour.

It is common intuition that as reward shaping directs exploration it can be both beneficial and detrimental to an agent's learning performance. If a good heuristic is used, common in previous published examples [7, 15, 24], the agent will learn quicker but the lesser published alternative is that a poor heuristic is used and the agent learns slower.[2]

However, the more important result of this example is to demonstrate that despite even the most misleading and never stable potential functions a single agent can still converge to the optimal policy. In the next section we go on to demonstrate a similar result but this time maintaining the guarantee of consistent Nash equilibria despite a never stable dynamic potential-function in a multi-agent problem domain.

4.2 Multi-Agent Example

To demonstrate consistent Nash equilibria with and without dynamic potential-based reward shaping, an empirical study of Boutilier's coordination game [4] will be presented here.

The game, illustrated in Figure 3, 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 3, each transition is labeled 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 multiple Nash equilibria; the joint policies opting for the safety state s_3 or the 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

[2]For single-agent examples of dynamic potential-based reward shaping providing beneficial gains in learning time we refer the reader to any existing published implementation [10, 11, 12, 13].

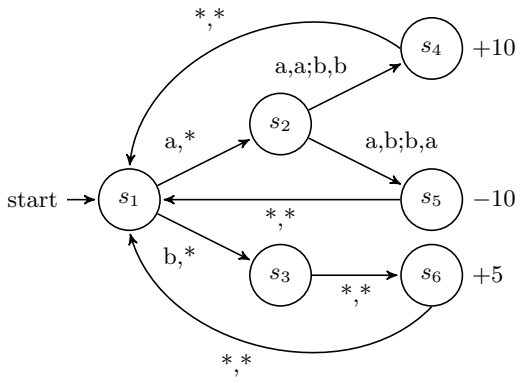


Figure 3: Boutilier's Coordination Game

state s_3.

As before we will compare the behaviour of agents with and without random dynamic potential-based reward shaping. Each agent will randomly assign its own potential to a new state upon entering it and be rewarded that potential discounted by γ less the potential of the previous state at the time it was entered. Therefore, each agent receives its own dynamic reward shaping unique to its own potential function. These experimental results are intended to show, that regardless of dynamic potential-based reward shaping, the shaped agents will only ever converge to one of the three original joint policy Nash equilibria.

The uniform random function will again choose potentials in the range 0 to 50. It is worthwhile to note here that, in this problem domain, the additional rewards from shaping will often be larger than those received from the environment when following the optimal policy.

The negative bias random function will choose potentials in the range 35 to 50 for state s_5 (the suboptimal state) or 0 to 15 for all other states. This potential function is bias towards the suboptimal policy, as any transition into state s_5 will be rewarded at least as high as the true reward for following the optimal policy.

All agents, both with and without reward shaping, use Q-learning with ϵ-greedy exploration and a tabular representation of the environment. Experimental parameters were set as $\alpha = 0.5$,$\gamma = 0.99$ and ϵ begins at 0.3 and decays by 0.99 each episode.

4.2.1 Results

All experiments were run for 500 episodes (15,000 action choices) and repeated 100 times. The results, illustrated in Figures 4, 5 and 6, plot the mean percentage of the last 100 episodes performing the optimal, safety and sub-optimal joint policies for the non-shaped and shaped agents. All figures include error bars illustrating the standard error from the mean. For clarity, graphs are plotted only up to 250 episodes as by this time all experiments had converged to a stable joint policy.

Figure 4 shows that the agents without reward shaping rarely (less than ten percent of the time) learn to perform the optimal policy. However, as illustrated by Figures 5 and 6, both sets of agents with dynamic reward shaping learn the optimal policy more often.

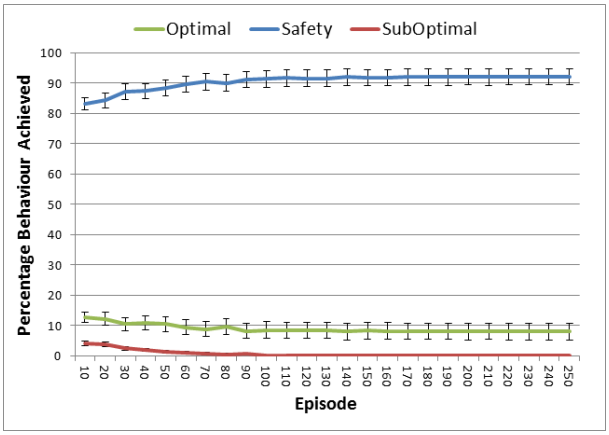


Figure 4: Without Reward Shaping

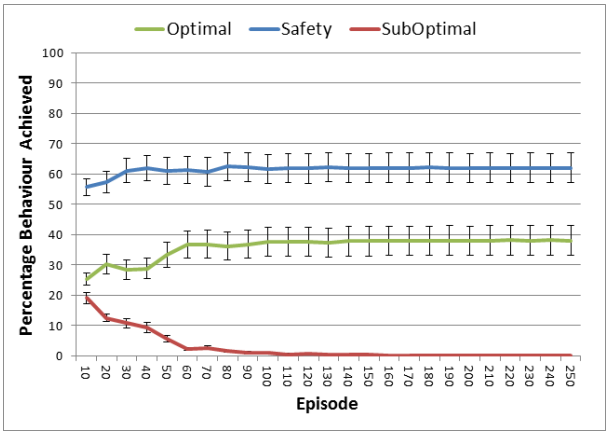


Figure 5: With Uniform Random Dynamic Reward Shaping

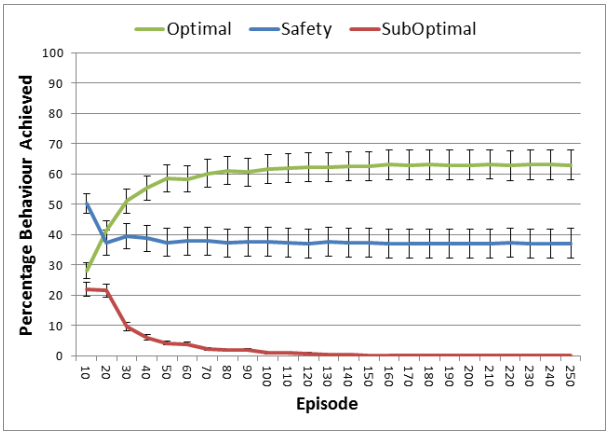


Figure 6: With Negative Bias Random Dynamic Reward Shaping

Therefore, in this domain, unlike the single-agent example, the dynamic reward shaping has been beneficial to final performance. This has occured because the agents' modified exploration has led to convergence to a different Nash equilibrium. However, please note, the agents never converge to perform the suboptimal joint policy. Instead the agents will only ever converge to the safety or optimal joint policies; the Nash equilibria of the unshaped and shaped systems. Thus demonstrating that even with dynamic reward transformations that never stabilise the Nash equilibria of the system remain the same provided the transformations are potential based.

5. CONCLUSION

In conclusion we have proven that a dynamic potential function can be used to shape an agent without altering its optimal policy provided the additional reward given is of the form:

$$F(s,t,s',t') = \gamma\Phi(s',t') - \Phi(s,t)$$

If multiple agents are acting in the same environment then, instead, the result becomes that the Nash equilibria remain consistent regardless of how many agents are receiving dynamic potential-based reward shaping.

Contrary to previous opinion, the dynamic potential function does not need to converge before the agent receiving shaping can as we have both theoretically argued and empirically demonstrated.

We have also proved that, although there is an equivalent Q-table initialisation to static potential-based reward shaping, it is not equivalent to dynamic potential-based reward shaping. We claim that no prior-initialisation can capture the behaviour of an agent acting due to a dynamic potential-based reward shaping as the changes that may occur are not necessarily known before learning begins.

Therefore, the use of dynamic potential-based reward shaping to inform agents of knowledge that has changed whilst they are learning is a feature unique to this method.

These results justify a number of pre-existing implementations of dynamic reward shaping [10, 11, 12, 13] and enable ongoing research into automated processes of generating potential functions.

6. ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their significant feedback and subsequent input to this paper.

7. 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 Seventh Annual International 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, 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. An empirical study of potential-based reward shaping and advice in complex, multi-agent systems. *Advances in Complex Systems*, 2011.

[8] S. Devlin and D. Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *Proceedings of The Tenth Annual International Conference on Autonomous Agents and Multiagent Systems*, 2011.

[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] M. Grześ and D. Kudenko. Online learning of shaping rewards in reinforcement learning. *Artificial Neural Networks-ICANN 2010*, pages 541–550, 2010.

[12] A. Laud. *Theory and application of reward shaping in reinforcement learning*. PhD thesis, University of Illinois at Urbana-Champaign, 2004.

[13] B. Marthi. Automatic shaping and decomposition of reward functions. In *Proceedings of the 24th International Conference on Machine learning*, page 608. ACM, 2007.

[14] M. Matarić. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4(1):73–83, 1997.

[15] 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.

[16] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1994.

[17] J. Randløv 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.

[18] 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.

[19] 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.

[20] R. S. Sutton. *Temporal credit assignment in reinforcement learning*. PhD thesis, Department of Computer Science, University of Massachusetts, Amherst, 1984.

[21] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[22] C. Watkins and P. Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[23] E. Wiewiora. Potential-based shaping and Q-value initialization are equivalent. *Journal of Artificial Intelligence Research*, 19(1):205–208, 2003.

[24] E. Wiewiora, G. Cottrell, and C. Elkan. Principled methods for advising reinforcement learning agents. In *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.

Learning and Predicting Dynamic Networked Behavior with Graphical Multiagent Models

Quang Duong[†] Michael P. Wellman[†] Satinder Singh[†] Michael Kearns[*]
{qduong,wellman,baveja}@umich.edu mkearns@cis.penn.edu
[†]Computer Science and Engineering, University of Michigan
[*]Computer and Information Science, University of Pennsylvania

ABSTRACT

Factored models of multiagent systems address the complexity of joint behavior by exploiting locality in agent interactions. *History-dependent graphical multiagent models* (hGMMs) further capture dynamics by conditioning behavior on history. The challenges of modeling real human behavior motivated us to extend the hGMM representation by distinguishing two types of agent interactions. This distinction opens the opportunity for learning dependence networks that are different from given graphical structures representing observed agent interactions. We propose a greedy algorithm for learning hGMMs from time-series data, inducing both graphical structure and parameters. Our empirical study employs human-subject experiment data for a dynamic consensus scenario, where agents on a network attempt to reach a unanimous vote. We show that the learned hGMMs directly expressing joint behavior outperform alternatives in predicting dynamic human voting behavior, and end-game vote results. Analysis of learned graphical structures reveals patterns of action dependence not directly reflected in the original experiment networks.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Multiagent Systems

General Terms

Experimentation, Algorithms, Human Factors

Keywords

graphical models, dynamic behavior, structure learning

1. INTRODUCTION

Modeling dynamic behavior of multiple agents presents inherent scaling problems due to the exponential size of any enumerated representation of joint activity. Even if agents make decisions independently, conditioning actions on each other's prior decisions or on commonly observed history induces interdependencies over time. To address this complexity problem, researchers have exploited the localized effects of agent decisions by employing *graphical models* of multiagent behavior. This approach has produced several (related) graphical representations capturing various facets of multiagent interaction [9, 11, 6, 5, 3]. *History-dependent graphical multiagent models* (hGMMs) [4] express multiagent behavior on an undirected graph, and capture dynamic relations by conditioning actions on history.

Prior work on hGMMs presumes a fixed graph structure defined by the modeler [4]. However, it is not always apparent how to choose the most salient inter-agent dependencies for accurate and tractable modeling. We seek methods for inducing hGMM structures from observational data about dynamic multiagent scenarios. In the process, we also extend the flexibility of hGMMs by allowing distinct dependence structures for within-time and across-time probabilistic relationships.

We empirically evaluate our techniques with data from laboratory experiments on dynamic consensus [8]. Human subjects were arranged on a network, specifying for each subject (also called *agent*) the set of others whose current choices are observable. The network associated with each experiment provides a basis for expecting that joint agent behavior may exhibit some locality that we can exploit in a graphical model for prediction.

We stress that the graph structure of the optimal predictive model need not mirror the experiment network of the voting scenario, and moreover, the complex experiment network instances we study render computation on the corresponding hGMMs intractable. Therefore, we attempt to learn the graphical structure and parameters of an hGMM that can effectively and compactly capture joint dynamic behavior. Using human subject data, we evaluate the learned models' predictions of voting behavior and compare their performance with those of different baseline multiagent models. We generally find that models expressing joint behavior outperform the alternatives, including models originally proposed by authors of the dynamic consensus experiments, in predicting voting dynamics. The joint behavior model provides comparable predictions on the rate of reaching consensus, and superior predictions of which consensus is reached. We further examine the learned hGMM graphical structures in order to gain insights about the dependencies driving voting behavior, as well as the network structure's effect on collective action.

Sections 2 provides background information on hGMMs, and introduces our extension to the modeling framework. Section 3 describes the dynamic consensus experiments. We present a variety of candidate model forms in Section 4. Sec-

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

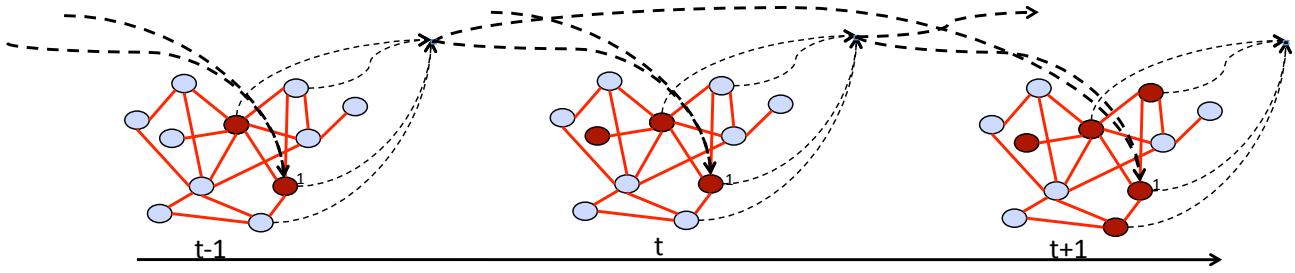


Figure 1: An example hGMM over three time periods. Undirected edges capture correlation among agents at a point in time. Directed edges (shown here only for agent 1) denote conditioning of an agent's action on others' past actions.

tion 5 provides motivations and details of our greedy model learning algorithm that simultaneously estimates a model's parameters and constructs its interaction graph. Our empirical study in Section 6 compares different models across three experiment settings, and examines the learned graph structures against the original experiment networks.

2. HISTORY-DEPENDENT GMMS

We model behavior of n agents over a time interval divided into discrete periods, $[0, \ldots, T]$. At time period t, agent $i \in \{1, \ldots, n\}$ chooses an action a_i^t from its action domain, A_i, according to its *strategy*, σ_i. Agents can observe others' and their own past actions up to time t, as captured in *history* $H^t = \{H_1^t, \ldots, H_n^t\}$, where H_i^t denotes the sequence of actions agent i has taken by t. Limited memory capacity or other computational constraints restrict an agent to focus attention on a subset of history H_i^t considered in its probabilistic choice of next action: $a_i^t \sim \sigma_i(H_i^t)$.

A *history-dependent graphical multiagent model* (hGMM) [4], $hG = (V, E, A, \pi)$, is a graphical model with graph elements V, a set of vertices representing the n agents, and E, edges capturing pairwise interactions between them. Component $A = (A_i, \ldots, A_n)$ represents the action domains, and $\pi = (\pi_1, \ldots, \pi_n)$ potential functions for each agent. The graph defines a neighborhood for each agent i: $N_i = \{j \mid (i,j) \in E\} \cup \{i\}$, including i and its neighbors $N_{-i} = N_i \backslash \{i\}$.

The hGMM representation captures agent interactions in dynamic scenarios by conditioning joint agent behavior on an abstracted history of actions H^t. The history available to agent i, $H_{N_i}^t$, is the subset of H^t pertaining to agents in N_i. Each agent i is associated with a potential function $\pi_i(a_{N_i}^t \mid H_{N_i}^t)$: $\prod_{j \in N_i} A_j \rightarrow R^+$. The potential of a local action configuration specifies its likelihood of being included in the global outcome, conditional on history. Specifically, the joint distribution of the system's actions taken at time t is the normalized product of neighbor potentials [2, 4, 7]:

$$\Pr(a^t \mid H^t) = \frac{\prod_i \pi_i(a_{N_i}^t \mid H_{N_i}^t)}{Z}. \quad (1)$$

The complexity of computing the normalization factor Z in (1) is exponential in the number of agents, and thus precludes exact inference and learning in large models. We approximate Z using the *belief propagation* method [1], which has shown good results with reasonable time in sparse cyclic graphical structures.

We extend the original hGMM representation by distinguishing between within-time and across-time dependencies, as depicted in Figure 1. Formally, we introduce a *conditioning set* Γ_i for each i, denoting the set of agents whose histories condition this agent's potential function: $\pi_i(a_{N_i}^t \mid H_{\Gamma_i}^t)$. The neighborhood N_i in this extension governs only the within-time probabilistic dependencies of node i. With respect to this extended model, the original hGMM [4] corresponds to the special case where $\Gamma_i = N_i$. The joint distribution of actions at time t can be rewritten as:

$$\Pr(a^t \mid H^t) = \frac{\prod_i \pi_i(a_{N_i}^t \mid H_{\Gamma_i}^t)}{Z}. \quad (2)$$

3. DYNAMIC CONSENSUS

We evaluate our modeling framework with human-subject data from a dynamic consensus game [8]. Each agent in this game chooses to vote either blue (0) or red (1), and can change votes at any time. Agents are connected in a network, such that agent i can observe the votes of those in its *observation neighborhood* N_i^O. The scenario terminates when: (i) agents converge on action $a \in \{0, 1\}$, in which case agent i receives reward $r_i(a) > 0$, or (ii) they cannot agree by the time limit T, in which case rewards are zero. Figure 2 illustrates the dynamic behavior of an example voting experiment network.

Agents have varying preferences for the possible consensus outcomes, reflected in their reward functions. As nobody gets any reward without a unanimous vote, agents have to balance effort to promote their own preferred outcomes against the common goal to reach consensus. Another important feature of the dynamic consensus game is that agent i observes the votes of only those in its observation neighborhood N_i^O, and all it is shown of the graph is the degree of each observation neighbor, and observation edges among them. This raises the question of how agents take into account their neighbors' voting patterns and their partial knowledge of the experiment network structure.

A series of human-subject experiments were conducted to study how people behave in 81 different instances of the voting game [8]. The experimenters varied reward preference assignments and experiment network structure in these experiment instances, and thus were able to collect data about these factors' effects on the consensus voting results, and the strategies employed. Figure 2 exhibits a run for the experimental network labeled power22, discussed below. Study goals included developing models to predict a given scenario's voting outcome, and if a consensus is reached, its convergence time. This problem also served as the founda-

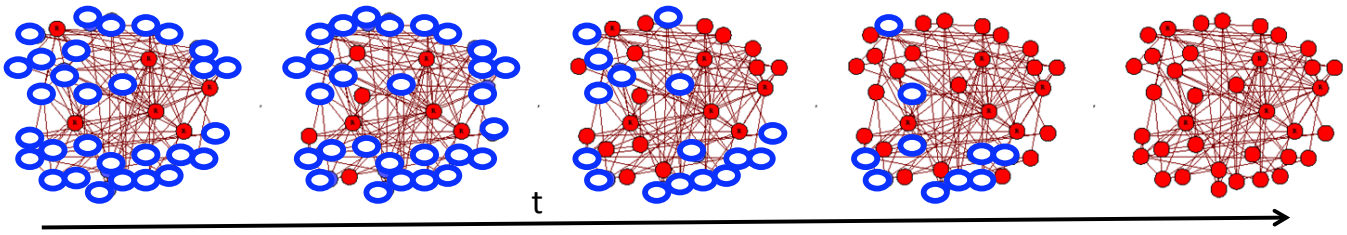


Figure 2: Time snapshots of a lab experiment run where the densely connected minority group that preferred red exerted strong influences on the blue-leaning majority. The minority group eventually succeeded in converting all the initial (unfilled) blue votes to (filled) red votes.

tion for analysis of adaptive strategies and theoretical constraints on convergence [10].

4. MODELING DYNAMIC VOTING

We present four model forms designed to capture voting behavior dynamics in the dynamic consensus experiments. All are expressible as hGMMs. Only the first exploits the flexibility of hGMMs to express dependence of actions within a neighborhood given history (2), hence we refer to this as the *joint behavior consensus model* (JCM).

The other three forms model agent behaviors individually: for each agent we specify a probabilistic strategy $\sigma_i(H^t) = \Pr(a_i^t \mid H^t_{\Gamma_i})$. Such a formulation captures agent interactions by the conditioning of individual behavior on observed history. The agents' actions are probabilistically dependent, but conditionally independent given this common history, yielding the joint distribution:

$$\Pr(a^t \mid H^t) = \prod_i \sigma_i(H^t). \tag{3}$$

We refer to a dynamic multiagent model expressible by (3) as an *individual behavior hGMM* (IBMM). Conditional independence given history is a compelling assumption for autonomous agents. Indeed, independent choice may even be considered definitional for autonomy. In practice, however, it is often infeasible to specify the entire history for conditioning due to finite memory and computational power, and the assumption may not hold with respect to partial history. History abstraction generally introduces correlations among agents actions, even if they are independently generated on full history [4]. Nevertheless, assuming conditional independence between agents' actions given history exponentially reduces the model's complexity, or more specifically, the representational complexity of the joint probability distribution over the system's actions.

The first of three IBMMs we present is designed as an independent behavior version of the JCM; thus, we call it simply the *individual behavior consensus model* (ICM). The remaining two models are based on proposals and observations from the original experimental analysis [8], and are labeled *proportional response model* (PRM) and *sticky proportional response model* (sPRM), respectively.

4.1 Joint Behavior Consensus Model

Based on observations from the original experiment analysis, we seek to formulate a potential function for JCM that captures the impact of past collective choices of i's neighborhood, i's own past voting patterns, and its relative preference for each action.

First, we consider how to summarize a history $H^t_{\Gamma_i}$ of length h relevant to agent i. Let indicator $I(a_i, a_k) = 1$ if $a_i = a_k$ and 0 otherwise. We define $f(a_i, H^t_{\Gamma_i})$ as the frequency of action a_i being chosen by other agents in i's conditioning set, which by definition contains nodes whose past influence how i chooses its action in the present,

$$f(a_i, H^t_{\Gamma_i}) = \frac{\sum_{k\in\Gamma_i\setminus\{i\}} \sum_{\tau=t-h}^{t-1} I(a_i, a_k^\tau) + \epsilon}{h|\Gamma_i \setminus \{i\}|}. \tag{4}$$

We add $\epsilon = 0.01$ to the numerator to ensure that the frequency term does not vanish when a_i does not appear in $H^t_{\Gamma_i}$.

Second, we capture agent i's own update history in an *inertia* term,

$$\mathcal{I}(a_i, H_i^t) = \begin{cases} t - \max_{\tau<t} \tau(1 - I(a_i^\tau, a_i)) & \text{if } a_i = a_i^{t-1} \\ [t - \max_{\tau<t} \tau(1 - I(a_i^\tau, a_i))]^{-1} & \text{otherwise} \end{cases}$$

In other words, we model agent i's voting inertia as proportional to how long it has maintained its most recent action a_i^{t-1}. We treat inertia as a factor tending to support the candidate of retaining the same action. If the candidate action a_i is different from a_i^{t-1}, the term expresses the inverse of this inertia factor.

Third, we define $r_i(a_{N_i})$ as the product of $r_i(a_i)$ and a heuristic attenuation based on how many nodes in the within-time neighborhood currently vote differently:

$$r_i(a_{N_i}) = \alpha^{\sum_{k\in N_i}(1-I(a_i,a_k))} r_i(a_i),$$

where $\alpha \in [0, 1]$. In our study, we set $\alpha = 0.9$. Observe that $r_i(a_{N_i})$ as defined is increasing in the number of i's neighbors voting a_i, reflecting the positive influence of neighbor choices on i.

The potential function for agent i combines these terms,

$$\pi_i(a_{N_i} \mid H^t_{\Gamma_i}) = r_i(a_{N_i}) f(a_i, H^t_{\Gamma_i})^\gamma \mathcal{I}(a_i, H_i^t)^\beta, \tag{5}$$

where $\gamma, \beta \geq 0$ denote the weight or importance of the historical frequency $f(a_i, H^t_{\Gamma_i})$ and the inertia $\mathcal{I}(a_i, H_i^t)$ relative to the estimated reward $r_i(a_{N_i})$. The normalized product of these potentials specifies joint behavior as described in (2). The model maintains two free parameters β and γ.

4.2 Individual Behavior Consensus Model

The ICM for dynamic consensus retains the main elements of JCM (5), while imposing conditional independence among agents' actions given the common history. The result is a within-time neighborhood N_i that contains only i itself for

each i. The probabilistic ICM behavior is then given by:

$$\Pr(a_i \mid H^t_{\Gamma_i}) = \frac{1}{Z_i} r_i(a_i) f(a_i, H^t_{\Gamma_i})^\gamma \mathcal{I}(a_i, H^t_i)^\beta.$$

The normalization ranges only over single-agent actions $a_i \in A_i$, thus Z_i is easy to compute for this model.

4.3 Proportional Response Model

We also consider for comparison the proportional response model, PRM, suggested in the original dynamic consensus study [8] as a reasonably accurate predictor of their experiments' final outcomes. PRM specifies that voter i chooses action a_i at time t with probability proportional to $r_i(a_i)$ and $g(a_i, a^{t-1}_{\Gamma_i})$, the number of i's neighbors who chose a_i in the last time period,

$$\Pr(a_i \mid H^t_{\Gamma_i}) \propto r_i(a_i) g(a_i, a^{t-1}_{\Gamma_i}).$$

4.4 Sticky Proportional Response Model

PRM does not capture the subjects' tendency to start with their preferred option, reconsidering their votes only after collecting additional information about their neighbors over several time periods [8]. Therefore, we introduce the *sticky proportional response model*, sPRM, which contains a parameter $\rho \in [-1, 1]$ reflecting an agent's stubbornness in maintaining its preferred option, regardless of observed neighbors' past choices. Intuitively, an agent's inherent bias toward its preferred option decays proportionally until there is no bias:

$$\Pr(a_i \mid H^t_{\Gamma_i}) \propto r_i(a_i) g(a_i, a^{t-1}_{\Gamma_i})(1 + \frac{I^{\max}_{a_i} \rho}{t}),$$

where $I^{\max}_{a_i} = 1$ if $a_i = \arg\max r_i(a)$ and $I^{\max}_{a_i} = 0$ otherwise.

5. LEARNING

5.1 Parameter Learning

We first address the problem of learning the parameters of an hGMM hG given the underlying graphical structure and data in the form of a set of joint actions for m time steps, $X = (a^0, \ldots, a^m)$. For ease of exposition, let θ denote the set of all the parameters that define the hGMM's potential functions. We seek to find θ maximizing the log likelihood of X,

$$L_{hG}(X; \theta) = \sum_{k=0}^{m-h} \ln(\Pr{}_{hG}(a^{k+h} \mid (a^k, \ldots, a^{k+h-1})); \theta)).$$

We use gradient ascent to update the parameters: $\theta \leftarrow \theta + \lambda \nabla\theta$, where the gradient is $\nabla\theta = \frac{\partial L_{hG}(X;\theta)}{\partial \theta}$, and λ is the learning rate, stopping when the gradient is below some threshold. We employ this same technique to learn the parameters of all model forms, except for the PRM which contains no parameters, in our study.

5.2 Structure Learning

Each of the consensus voting experiments involves 36 human subjects. The largest neighborhood size in these games ranges from 16 to 20, rendering computing exact data likelihood for a joint behavior model of this complexity (required for parameter learning described above) infeasible. Preliminary trials with the belief propagation approximation algorithm [1] on these models, where $N = \Gamma = N^{\mathsf{O}}$, indicated that this computational saving would still be insufficient for effective learning. Thus, we need to employ models with simpler graphs in order to take advantage of hGMMs' expressiveness in representing joint behavior. Toward this end, we developed a structure learning algorithm that produces graphs for hGMMs within specified complexity constraints.

Though dictated by computational necessity, automated structure learning has additional advantages. First, there is no inherent reason that the observation graph should constitute the ideal structure for a predictive graphical model for agent behavior. In other words, the most effective N is not necessarily the same as N^{O}. Since actual agent behavior is naturally conditioned on its observable history, we do assume that the *conditioning set* coincides with the observation neighborhood, $\Gamma = N^{\mathsf{O}}$. Nevertheless, once we abstract the history representation it may well turn out that non-local historical activity provides more useful predictive information. If so, the structure of the learned graph that defines each i's within-time neighborhood may provide interesting insights on the agents' networked behavior.

Our structure learning algorithm addresses the problem of learning N_i for every i, taking $\Gamma_i = N^{\mathsf{O}}_i$ as fixed. Note that the IBMMs described in Section 4 impose $N_i = \{i\}$ for each i, and thus do not need to learn the within-time graphs. Starting from an empty graph, we greedily add edges to improve the log-likelihood of the training data, subject to a constraint that the maximum node degree not exceed a specified bound $d_{\max}$. Since the set of edges E is the only structural model feature that changes during our search, we use $L_E(X; \theta)$ to abbreviate $L_{hG}(X; \theta)$ as induced by the hGMM $hG = (V, E, A, \Gamma, \pi)$. We have found that the optimal settings of our parameters $\theta = (\beta, \gamma)$ is insensitive to within-time dependencies, hence we apply the parameter learning operation (Section 5.1) only once, at the beginning of our search. The algorithm is defined formally below.

$E \leftarrow \emptyset$
Use gradient descent to identify $\theta \approx \arg\max L_E(X; \theta)$.
$\tilde{E} \leftarrow \{(i, j) \mid i \in V, j \in V\}$
repeat
  newedge $\leftarrow$ **false**
  $(i_*, j_*) \leftarrow \arg\max_{(i,j) \in \tilde{E}} L_{E \cup (i,j)}(X; \theta)$
  if $L_{E \cup (i_*, j_*)}(X; \theta) \geq L_E(X; \theta)$ **then**
    $E \leftarrow E \cup \{(i_*, j_*)\}$
    newedge $\leftarrow$ **true**
  end if
  $\tilde{E} \leftarrow \tilde{E} \setminus \{(i_*, j_*)\} \setminus \{(i, j) \mid \max(|N_i|, |N_j| = d_{\max}\}$
until $\tilde{E} = \emptyset \vee$ *newedge* = **false**

5.3 Evaluation

We evaluate the learned multiagent models by their ability to predict future outcomes, as represented by a test set Y. Given two models M_1 and M_2, we compute their corresponding log-likelihood measures for the test data set Y: $L_{M_1}(Y)$ and $L_{M_2}(Y)$. Note that since log-likelihood is negative, we instead examine the negative log-likelihood measures, which means that M_1 is better than M_2 predicting Y if $-L_{M_1}(Y) < -L_{M_2}(Y)$, and vice versa.

6. EMPIRICAL STUDY

We empirically evaluate the predictive power of JCMs in comparison with ICMs, PRMs, and sPRMs, using the dy-

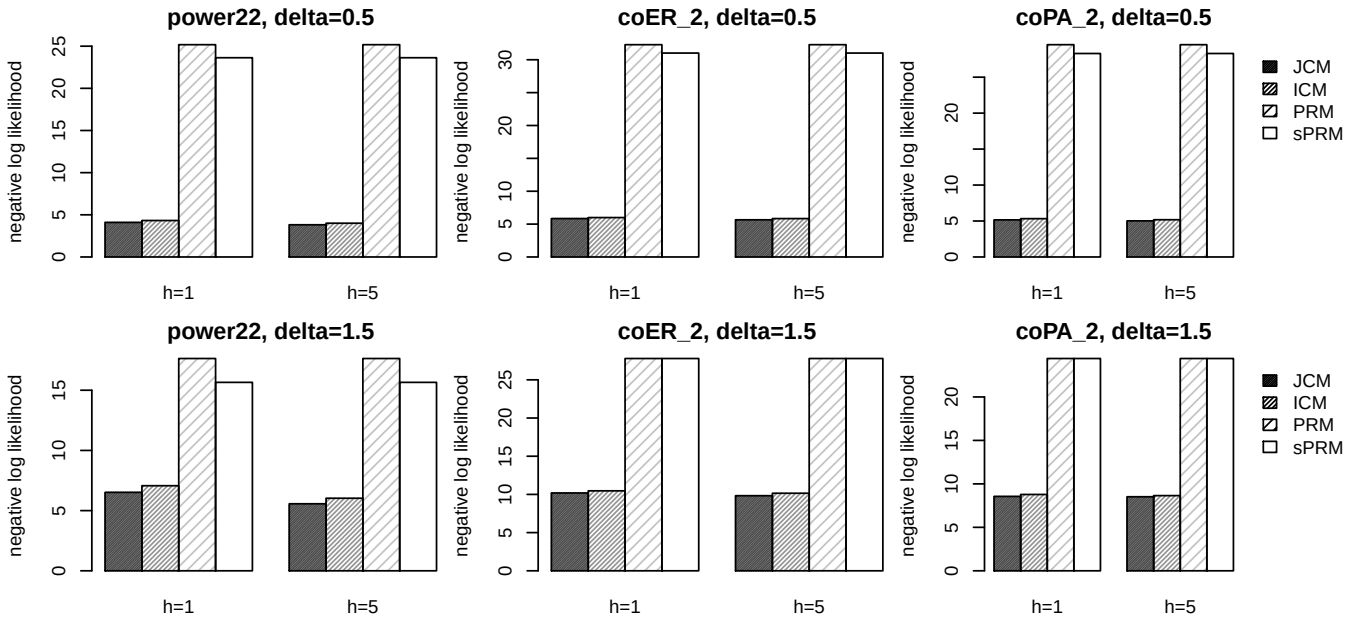


Figure 3: JCMs provide better predictions of the system's dynamics than ICMs, PRMs, and sPRMs in twelve settings: the three experiment networks power22 (left), coER_2 (middle), and coPA_2 (right), each for two history lengths, using time discretization intervals $\delta = 0.5$ (top) and $\delta = 1.5$ (bottom). The prediction quality differences between JCM and ICM are significant ($p < 0.025$) in all scenarios.

namic consensus experiment data [8]. We also examine the graphs induced by structure learning, and relate them to the corresponding observation networks by various statistical measures.

6.1 Experiment Settings

The human-subject experiments are divided into nine different sets, each associated with a network structure. These structures differ qualitatively in various ways, characterized by node degree distribution, ratio of inter-group and intra-group edges, and the existence of a well-connected minority [8]. In particular, networks whose edges are generated by a random Erdos-Renyi (ER) process have a notably more heavy-tailed degree distribution than those generated by a preferential attachment (PA) process. For each experimental trial, human subjects were randomly assigned to nodes in the designated network structure, and preferences based on one of three possible incentive schemes. Since subjects in these experiments can change their votes at any time, the resulting data is a stream of asynchronous vote actions. We discretize these streams for data analysis, recording the subjects' votes at the end of each time interval of length δ seconds. Our experiments examine interval lengths $\delta \in \{0.5, 1.5\}$.

In our study, we learn predictive models for each experiment network structure, pooling data across subject assignments and incentive schemes. This approach is based on the premise that network structure is the main factor governing the system's collective behavior, in line with the original study findings [8]. In each experiment set, we use eight of the nine trials for training the predictive models for each form. The within-time graphs are learned with node degree constraint $d_{\max} = 10$. We then evaluate the models based on their predictions over a test set comprising the left-out experimental trial. This process is repeated five times, with a different randomly chosen trial reserved for testing. Each data point in our reported empirical results averages over these five repetitions.

Using the original experiment labels, we distinguish three experiment networks according to their graph generator processes and the existence of a minority group of well-connected nodes that share the same vote preference (see Table 1).

Table 1: Voting Experiment Settings

Label	Strong Minority	Graph Generator Process
coER_2	No	Erdos-Renyi
coPA_2	No	Preferential attachment
power22	Yes	Preferential attachment

6.2 Predictions

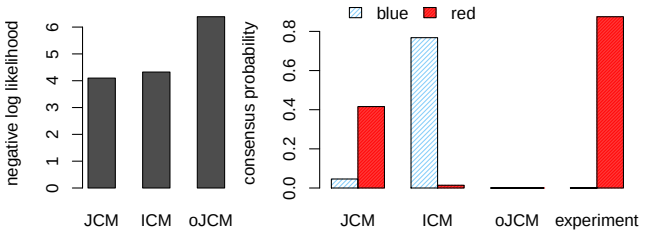


Figure 5: oJCMs provide worse predictions than JCMs and ICMs for both the system's dynamics and end-game results (power22, $h = 1$ and $\delta = 0.5$).

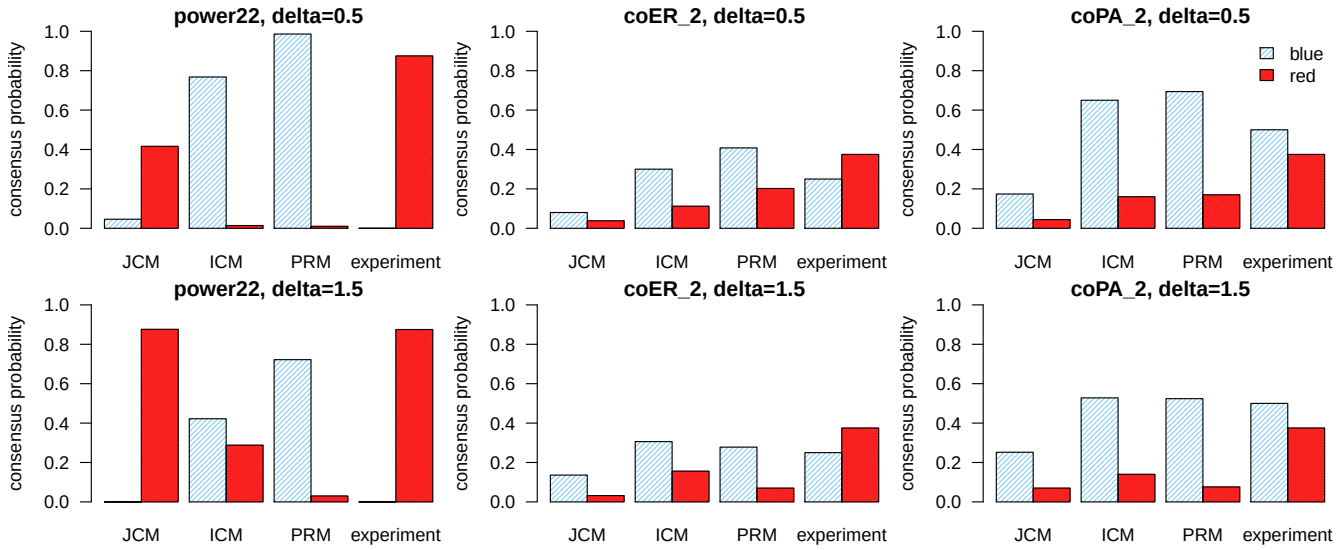


Figure 4: JCM predictions on the probability of reaching consensus are lower than predictions from ICMs and PRMs, as well as experiment outcomes. However, the JCM is significantly more accurate than ICMs or PRMs on predicting the ultimate consensus colors.

We first examine predictions of subjects' votes in each time period conditional on available history. A comparison of four models on twelve scenarios is presented in Figure 3. We measure predictive performance by negative log-likelihood of the test data, according to the respective models. JCMs perform better than ICMs, PRMs, sPRMs, in predicting dynamic behavior in the dynamic consensus experiments for all three experiment settings, given data discretized at interval lengths of 0.5 and 1.5 (differences significant at $p < 0.025$). Both the JCM and ICM representations, which share similar fundamental elements, handily outperform PRM and its sticky version sPRM.

Contrary to the expectation that the less historical information a model uses, the lower its prediction performance, JCMs and ICMs that employ only the last $h = 1$ period of historical data generate similar predictions to those with $h = 5$. This phenomenon is likely a consequence of the heuristic nature of the frequency function (4), and moreover may indicate that some human subjects take into account only a short history of their neighbors' actions when choosing their own actions. All models perform worse with the larger time interval $\delta = 1.5$, which is unsurprising in that the coarser discretization entails aggregating data. More salient is that the results are qualitatively identical for the two δ settings, further illustrating the robustness of our findings. These results in general demonstrate JCMs' ability to capture joint dynamic behavior, especially behavior interdependencies induced by limited historical information, as opposed to the IBMM alternatives.

We next evaluate the models' ability to predict the end state of a dynamic consensus experiment. As noted above, the original aim of modeling in these domains was to predict this final outcome. For a particular model M, we start a simulation run with agents choosing their preferred colors, and then proceed to draw samples from M for each time period until a consensus is reached or the number of time periods exceeds the time limit. We average over 100 run instances for each environment setting and model. As we do not observe any considerably qualitative differences in the models' end-game predictions for different history lengths h, we choose to display only results for $h = 1$ henceforth.

The proportion of simulation instances reaching consensus induced by ICMs and PRMs correlates with observed experiment results, as shown in Figure 4.[1] Simulated runs drawn from JCMs converge to consensus at lower rates than in ICMs, PRMs, and human-subject experiments in general. However, their end-game predictions improve with greater $\delta = 1.5$, especially in the power22 setting where JCMs predict the experiment outcomes almost exactly. A closer look at the end-game prediction results reveals a different picture about the relative performances of the three models. In particular, the individual behavior models' predictions on the final consensus color are considerably out of line with the actual experiments for both coER_2 and power22, rendering them ineffective in predicting end-game color results. JCMs, on the other hand, provide significantly more accurate predictions on the consensus color in the power22 setting. The ratio between blue and red consensus instances by JCMs in coPA_2 resembles that of the actual experiments more than ICMs and PRMs. In the coER_2 setting all models' predictions on the favored consensus color (blue) miss the actual experiments' favored consensus color (red), though the ratio of red-to-blue consensus predicted by JCM is less skewed than that of ICMs and PRMs.

Last, we demonstrate the benefits of our extension to the original hGMM representation by comparing the JCM representation against oJCM, which retains the original hGMM definition, assuming that the conditioning set is identical to the learned within-time neighborhood: $\Gamma = N$. Figure 5 shows that oJCMs perform worse than both JCMs and ICMs in predicting the system's votes for each time period and end-game results, for the power22 setting with $h = 1$ and

[1]End-game results from sPRMs are similar to those from PRMs, and not shown here.

$\delta = 0.5$.[2] Moreover, we note that the resulting graphs by oJCMs contain disconnected node subsets, which potentially prevent vote decisions to propagate throughout the network, causing failures in producing any consensus instances.

6.3 Graph Analysis

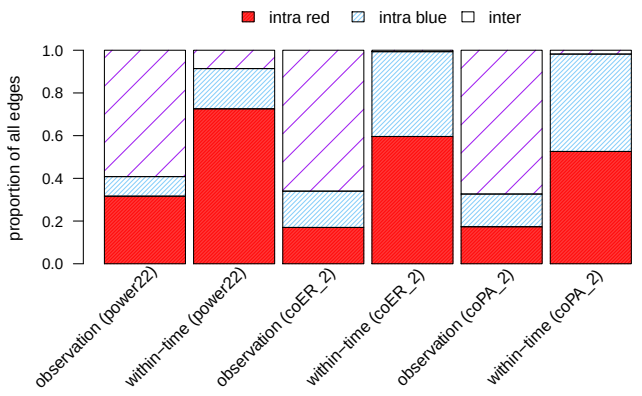


Figure 6: Distributions of edges from three different categories, intra red, intra blue, and inter, in the given observation and learned within-time graphs for JCM ($\delta = 0.5$).

In this section, we seek to characterize the learned edges that define N in the JCM representation, and discover connections between the learned graphs and the aforementioned prediction results. First, we categorize edges by their endpoint nodes' vote preferences: we refer to edges that connect two red (blue) nodes as *intra* red (blue), and those between red and blue nodes as *inter* edges. Figure 6 presents the proportion of each edge type in both the given observation graphs and the learned within-time graphs. While a majority of edges in the observation graphs are inter edges, the within-time graphs that define N consist mostly of intra edges. That is, there are more interdependencies in JCMs among agents of the same preference than among conflicting agents. The ability to discover these inter edges and incorporate the information they carry in its joint action distribution may help the JCM representation to better capture dynamic behavior and end-game results, as illustrated and discussed in Section 6.2. For the power22 setting in particular, JCMs often assign a majority of edges as intra red, and thus effectively identify the presence of a strongly connected red minority who dictated end-game colors in the actual experiments. This construction allows JCMs to predict end-game consensus colors much more accurately than ICMs and PRMs, which rely entirely on the observation graphs.

We further investigate whether these proportion measures provide any predictions on the number of consensus instances induced by JCMs. We pool data from the three experiment settings—power22, coPA_2, and coER_2—and compute a simple linear regression of the number of red (blue) consensus instances with respect to the proportion of intra red (blue) edges. The resulting regression coefficients are statistically significant for both blue and red ($p < 0.05$).

[2]We also obtain similar results for oJCMs in other experiment settings and environment parameters, which are not shown here.

Figure 7 suggests that a weak positive correlation between the within-time graphs' intra edges and the number of consensus instances. Intuitively, more interdependence between same-preference nodes allows them to have more influence on one another, helping to diffuse vote choices more rapidly throughout the system.

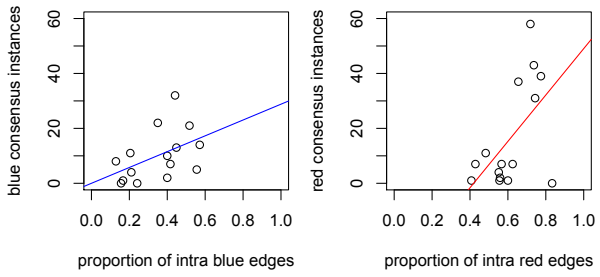


Figure 7: The number of consensus instances in blue (left) and red (right), and proportion of JCM intra edges of the corresponding colors.

We next examine JCM edges in terms of how far apart are the nodes they connect in the observation graph. Let $\phi_{i,j} \geq 1$ denote the length of shortest path from i to j in the observation graph. Given a graph G on the same set of nodes, we can calculate the proportion of edges in G that connect nodes separated by a certain distance in the original observation graph. Figure 8 presents the profile of such distances for pairs of nodes in the learned JCMs. For comparison, the profiles labeled "fully connected" simply reflect the distribution of node distances in the original observation graph: most of the nodes are one hop or less apart from each other ($\phi \leq 2$), and the modal distance is $\phi = 2$. A large majority of edges in the learned within-time graphs have $\phi = 2$, that is, are close but not connected in the observation graphs.

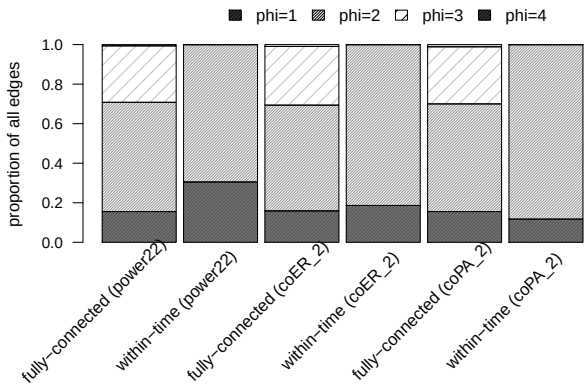


Figure 8: Distributions of edges in the within-time graphs based on the distance between their end-nodes ϕ in the observation graph ($\delta = 0.5$).

Next we compare the *assortativity* [12] of the learned and original graphs. A graph G's assortativity coefficient in $[-1, 1]$ captures the tendency for nodes to attach to others that are similar (positive values) or different (negative values) in connectivity. As illustrated in Figure 9, the large difference in assortativity for the power22 setting stresses the

JCM's ability to discover interdependencies among agents' actions that are not captured in the observation graph. In particular, the resulting JCMs are able to capture action correlations among nodes of similar degrees in the power22 setting, where the minority nodes are more densely connected than the majority, confirming the findings by aforementioned graph analyses on intra and inter edges. We also investigate the sparsity of the learned graphs for different values of δ. Our sparsity measure is the number of edges in the learned within-time graph divided by the number of edges in the corresponding observation graph. Figure 9 illustrates that the within-time graphs become sparser as the discretization interval shrinks from 1.5 to 0.5 in all experiment settings. Intuitively, the finer grain the discretization is, the fewer simultaneous vote changes there are in one time period. As a result, there may be fewer interdependencies among agents' actions, which explains the aforementioned relations between discretization interval and graph sparsity across all experiment settings.

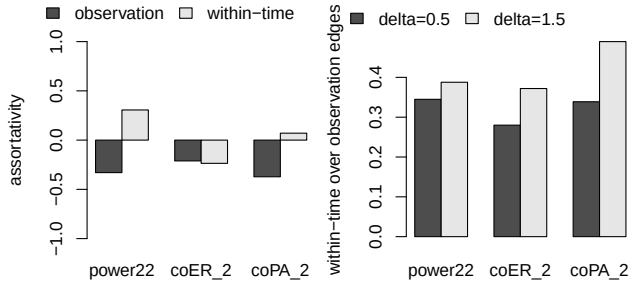


Figure 9: (left) Assortativity of the observation graphs and the learned within-time graphs ($\delta = 0.5$). (right) Sparsity of the within-time graphs.

7. CONCLUSIONS

Our main result is a demonstration of the feasibility of learning probabilistic models of dynamic multiagent behavior from real traces of agent activity on a network. To accomplish this we extend the original hGMM framework [4], by distinguishing within-time dependencies from conditioning sets, and introducing a structure-learning algorithm to induce these dependencies from time-series data. We evaluated our techniques by learning compact graphs capturing the dynamics of human-subject voting behavior on a network. Our investigation finds that the learned joint behavior model provides better predictions of dynamic behavior than several individual behavior models, including the proportional-response models suggested by the original experimental analysis. This provides evidence that expressing joint behavior is important for dynamic modeling, even given partial history information for conditioning individual behavior. Our graph analysis further reveals characteristics of the learned within-time graphs that provide insights about patterns of agent interdependence, and their relation to structure of the agent interaction network.

We plan to improve the learning algorithm for individual behavior models, by replacing the maximum-degree constraint with a cross-validation condition that can better help avoid over-fitting. Given the formalism's generality, we consider it promising to apply our modeling technique to similar problem domains, such as graph coloring, where agents must coordinate their actions or make collective decisions while only communicating with their neighbors, as well as large network scenarios, such as social networks and Internet protocols.

8. REFERENCES

[1] J. Y. Broadway, J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. In *Thirteenth Annual Conference on Advances in Neural Information Processing Systems*, pages 689–695, Denver, 2000.

[2] C. Daskalakis and C. H. Papadimitriou. Computing pure Nash equilibria in graphical games via Markov random fields. In *Seventh ACM conference on Electronic Commerce*, pages 91–99, Ann Arbor, MI, 2006.

[3] Q. Duong, M. P. Wellman, and S. Singh. Knowledge combination in graphical multiagent models. In *Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 153–160, Helsinki, 2008.

[4] Q. Duong, M. P. Wellman, S. Singh, and Y. Vorobeychik. History-dependent graphical multiagent models. In *Ninth International Conference on Autonomous Agents and Multiagent Systems*, pages 1215–1222, Toronto, 2010.

[5] Y. Gal and A. Pfeffer. Networks of influence diagrams: A formalism for representing agents' beliefs and decision-making processes. *Journal of Artificial Intelligence Research*, 33:109–147, 2008.

[6] A. X. Jiang, K. Leyton-Brown, and N. A. R. Bhat. Action-graph games. *Games and Economic Behavior*, 71:141–173, 2010.

[7] S. Kakade, M. Kearns, J. Langford, and L. Ortiz. Correlated equilibria in graphical games. In *Fourth ACM Conference on Electronic Commerce*, pages 42–47, San Jose, CA, 2003.

[8] M. Kearns, S. Judd, J. Tan, and J. Wortman. Behavioral experiments on biased voting in networks. *Proceedings of the National Academy of Sciences*, 106(5):1347–1352, 2009.

[9] M. Kearns, M. L. Littman, and S. Singh. Graphical models for game theory. In *Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 253–260, Seattle, 2001.

[10] M. Kearns and J. Tan. Biased voting and the Democratic primary problem. In *Fourth International Workshop on Internet and Network Economics*, pages 639–652, Shanghai, 2008.

[11] D. Koller and B. Milch. Multi-agent influence diagrams for representing and solving games. *Games and Economic Behavior*, 45:181–221, 2003.

[12] M. E. J. Newman. Mixing patterns in networks. *Physical Review E*, 67(2), 2003.

Session 3C
Human-agent Interaction

A Cultural Sensitive Agent for Human-Computer Negotiation

Galit Haim
Bar Ilan University, Israel
haimga@cs.biu.ac.il

Ya'akov (Kobi) Gal
Ben-Gurion University of the Negev, Israel
kobig@bgu.ac.il

Sarit Kraus*
Bar Ilan University, Israel
sarit@cs.biu.ac.il

Michele Gelfand
University of Maryland, USA
mgelfand@umd.edu

ABSTRACT

People's cultural background has been shown to affect the way they reach agreements in negotiation and how they fulfill these agreements. This paper presents a novel agent design for negotiating with people from different cultures. Our setting involved an alternating-offer protocol that allowed parties to choose the extent to which they kept each of their agreements during the negotiation. A challenge to designing agents for such setting is to predict how people reciprocate their actions over time despite the scarcity of prior data of their behavior across different cultures. Our methodology addresses this challenge by combining a decision theoretic model with classical machine learning techniques to predict how people respond to offers, and the extent to which they fulfill agreements. The agent was evaluated empirically by playing with 157 people in three countries—Lebanon, the U.S., and Israel—in which people are known to vary widely in their negotiation behavior. The agent was able to outperform people in all countries under conditions that varied how parties depended on each other at the onset of the negotiation. This is the first work to show that a computer agent can learn to outperform people when negotiating in three countries representing different cultures.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]

General Terms

Experimentation

Keywords

Human-robot/agent interaction, Negotiation

1. INTRODUCTION

The dissemination of technology across geographical and ethnic borders is opening up opportunities for computer agents to negotiate with people of diverse cultures and backgrounds. For example, electronic commerce (e.g., ebay), crowd-sourcing (e.g., Amazon Turk) and deal-of-the-day applications (e.g., Groupon) already involve computer agents that make decisions together with people from different countries. People's cultural background has been shown to be a key determinant of the way they make and keep their agreements with others [7]. It is thus important for agent designers to model how people from various cultures respond to different kinds of decision-making behavior employed by others. To succeed in such settings computer agents need to adapt to the culture and particular behavior of the individual they interact with.

This paper presents a novel agent-design for settings in which participants repeatedly negotiate over the exchange of scarce resources, and agreements are not binding. Such settings characterize the real-world applications shown above, in that participants make commitments to purchasing items or carrying out tasks, they can choose whether and how to fulfill these commitments, and these decisions affect their future interactions with the other participants. For example, a seller that delivers an item very late to a buyer, or does not deliver an item at all, may be negatively reciprocated by the buyer in a future transaction.

Prior work has addressed some of the computational challenges arising in repeated negotiation between people and computer agents [6, 11]. However, additional challenges arise when designing agents that adapt to different cultures. First, agents need to adopt a separate strategy in each culture, requiring large amounts of data to be collected of people's play in the different cultures. Second, people's individual behavior within a culture displays wide variance, as people's strategies are inconsistent and prone to noise [7]. Thus, a computer agent needs to adapt quickly to the individual strategy of its negotiation partner over time. To address the first challenge, we combined a decision-theoretic model with classical machine learning techniques to model human behavior in different cultures. The decision-theoretic model used an influence diagram to efficiently represent and reason about the negotiation process. The learning techniques were based on features that represent players' states in the negotiation, as well as social factors that reflect their generosity and reliability over time. To address the second challenge, we collected data in three different countries in which people are known to exhibit distinct cultural differences in their

*also affiliated with the University of Maryland Institute for Advanced Computer Studies.

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

negotiation behavior (Israel, Lebanon and the U.S). In order to boost the amount of data available for learning, we trained the model of people's play with baseline computer agents as well as other people.

Our proposed agent incorporated the learned models of people's play and solved the influence diagram to make decisions in the game. We evaluated this agent when negotiating with new people in each of the three countries. Our results show that the agent was able to outperform people in all three countries, and in conditions varying how participants depended on each other's resources during negotiation. The agent learned to adapt to the behavior of its negotiation partner over time in each of the cultures, and used a general model of their behavior when little or no history of play was available. These results demonstrate the need for agent-designers to model the effects of culture on human behavior when agents are deployed globally or in multi-cultural settings. It is the first work to show that a computer agent can learn to outperform people when negotiating in three different cultures.

There is a body of work in the psychological and social sciences that investigates cross-cultural behavior among human negotiators [2]. However, there are scant computational models of human negotiation behavior that reason about cultural differences. Past work in AI has used machine learning and opponent modeling approaches toward building computer agents that negotiate with people. [13].Jonker et al. [10] designed an agent architecture that used concession strategies to avoid impasses in the negotiation. Byde et al. [1] constructed agents that bargain with people in a market setting by modeling the likelihood of acceptance of a deal and allowing agents to renege on their offers. Oshrat et al. [12] have used learning techniques to model the extent to which people exhibit different social preferences when they accept offers in one-shot and multiple interaction scenarios. To date, all work on human-computer negotiation assumes that agreements are binding and have relied on prior data of people's negotiation behavior. A notable exception is the work by Gal et al. [5] that proposed an agent for negotiating with people in the U.S. and Lebanon using the same protocol and setting as this work. However, this agent used hand-designed rules of behavior and did not model its partner in a formal way. This agent was able to outperform people in the U.S. but not in Lebanon, whereas our learned agent was able to outperform people in the U.S., Lebanon and Israel. Thus, our work is novel in showing that combining decision theory and machine learning is a better approach towards building agents in the same settings they considered.

2. IMPLEMENTATION: COLORED TRAILS

Our empirical setting consisted of a game that interleaved negotiation to reach agreements and decisions of whether and how much to fulfil the agreement. The game was configured using the Colored Trails (CT) game [4] and played on a 7x5 board of colored squares. One square on the board was designated as the goal square. Each player's icon was initially located in one of the non-goal positions, eight steps away from the goal square. To move to an adjacent square, a player needed to surrender a chip in the color of that square.

At the onset of the game, one of the players was given the role of proposer, while the other was given the role of responder. The interaction proceeded in a recurring sequence of phases, using an alternating offers protocol. In the "negotiation phase", the player designated as the proposer could make an offer to the other player, who was designated the responder. In turn, the responder could accept or reject the offer. If the offer was rejected, then players switched roles: the responder became the proposer and the proposer became the responder. After the counter offer was accepted or rejected by the responder, the game moved to the next phase. In the "transfer phase" both players could choose chips to transfer to each other. The transfer action was done simultaneously, such that neither player could see what the other player transferred until the end of the phase. A player could choose to transfer more chips than it agreed to, or any subset of the chips it agreed to, including transferring no chips at all. In the "movement phase" both players could move their icons on the board one step towards the goal square, provided they had the necessary chip. Players alternated their roles, such that the first proposer in the previous negotiation phase was designated as a responder in the next negotiation phase, and vice versa. These phases repeated until the game ended, which occurred when one of the following conditions held: (1) at least one of the participants reached the goal square; or (2) at least one of the participants remained dormant and did not move for three movement phases.

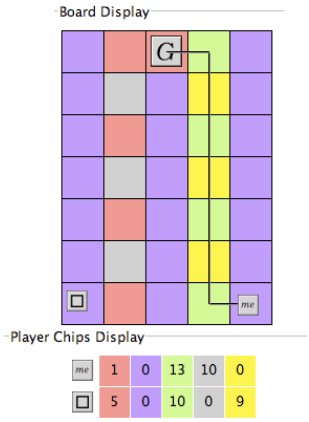


Figure 1: An example of a CT Board

When the game ended, both participants were automatically moved as close as possible to the goal square, and their score was computed as follows: 100 bonus points for getting to the goal square, 5 bonus points for any chip left in a player's possession; a 10 point penalty for each square left in the path from a player's final possession to the goal square. These parameters were chosen so that getting to the goal was by far the most important component, but if a player could not get to the goal it was preferable to get as close to the goal as possible. Note that players had full view of the board and each others' chips, and thus they had complete knowledge of the game situation at all times during the negotiation process.

An advantage of using CT is that it provides a realistic analog to task settings, highlighting the interaction among goals, tasks required to achieve these goals and resources needed for completing tasks. In CT, chips correspond to agent capabilities and skills required to fulfill tasks. Different squares on the board represent different types of tasks. A player's possession of a chip of a certain color corresponds to having the skill available for use at that time.

We used two different types of boards in the study to represent different dependency relationships between players. In one of the boards, neither player could reach the goal given its initial chip allocation, and there existed at least one exchange such that both players could reach the goal. We referred to players in this game as *task co-dependent*. In the other board type, one of the players, referred to *task independent*, possessed the chips it needed to reach the goal, while the other player, referred to as *task dependent*, required chips from the task-independent player to get to the goal. An example of the co-dependent board used in our study is shown in Figure 1. In this game both the "me" icon and "square" icon players were missing three chips to get to the goal: The "me" player was missing three yellow chips whereas the "square" player was missing three gray chips. The relevant path from the point of view of the "me" player is outlined.

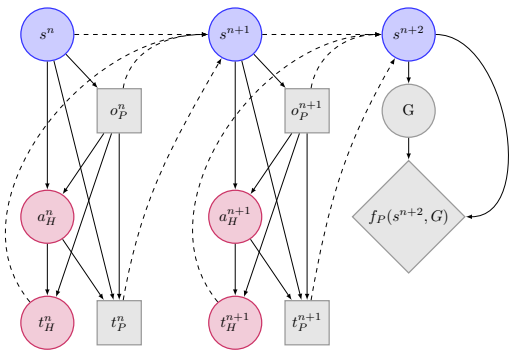


Figure 2: An Influence Diagram for two rounds of interaction in the CT game.

3. A DYNAMIC MODEL OF INTERACTION

In this section we describe an agent-design termed the Personality Adaptive Learning (PAL) agent. Before describing the decision theoretic model used by PAL, we make the following definitions. Let n denote an arbitrary negotiation phase in the game. For any two participants i and j, let c_i^n denote the set of chips in possession of i at phase n in the game. Let $o^n = (o_i^n, o_j^n)$ denote a proposal at round n in which $o_i^n \subseteq c_i^n$ is the set of chips that i sends to j, and o_j^n is the set of chips that j sends to i. Note that the proposal o^n can be made by either player i or player j. Let a^n denote the other player's response to o^n whether to reject or accept the proposal. Let $t_i^n \subseteq c_i^n$ be the set of chips transferred by i following the response, and let $t_j^n \subseteq c_j^n$ be the set of chips transferred by j. The protocol allows participants to transfer chips regardless of whether or not an offer is accepted.

The *current score* for i at round n measures the score in the game given its chips c_i^n. This is defined as $u_i^n(c_i)$.[1] Given a proposal o^n at round n, the *promised score* to player i at round n measures the score in the game that i would receive in the case that j was fully reliable and sent o_j^n promised chips to player i. The *reliability measure* of player j at round n, denoted r_j^n, reflects the extent to which j fulfilled its commitment to send o_j^n chips to i. It is defined as the ratio between the current score to i after j transferred chips and the promised score to i. A reliability measure of 1 means that player j transferred all of its promised chips to i; the extent to which the reliability measure is lower than 1 represents the degree to which the player did not fulfill its commitment for a given agreement, as defined below.

$$r_j^n = \frac{u_i(c_i^n \cup t_i^n)}{u_i(c_i^n \cup o_j^n)} \quad (1)$$

Note that the reliability measure of j only depends on the chips it sent to i, and does not depend on the chips sent by i to j. The reliability of player i is defined symmetrically, and omitted for brevity.

PAL uses an influence diagram [9] to efficiently represent and reason about its decisions over time. An influence diagram is a directed acyclic graph containing three kinds of nodes: chance nodes denoted by ellipses, decision nodes denoted by gray rectangles, and utility nodes denoted by diamonds. Each chance node has an associated conditional probability distribution (CPD). A utility node has an associated deterministic function from values of its parents to the real numbers. The parents of a decision node represent information that is known to the decision maker at the time of making the decision, and are called informational parents. Each decision and utility node is associated with a particular agent. An influence diagram representing two rounds of interaction in the game is shown in Figure 2. Each decision node is labeled with the corresponding decision for PAL in the game and appears in gray background. The decision nodes o_P^n and t_P^n represent the proposal made by PAL and the chips it decides to transfer at round n. Similarly, the decision node o_P^{n+1} and t_P^{n+1} represent the proposal made by PAL and the chips that it decides to transfer at round $n+1$.

The person's decisions in the influence diagrams are modeled as chance nodes. The nodes a_H^n and a_H^{n+1} represent the person's response to the proposal made by PAL in rounds n and $n+1$. The node t_H^n and t_H^{n+1} represents the person's decision to transfer chips at round $n+1$.

The node s^n represents the state of the game at round n. This node is a parent of all decisions made by both participants in round n. This represents the fact that the state is observed by the participants in the game when making their decisions in each round. The state encapsulates the history of the game into a tuple that contains relevant information about the game, as well players' reliability measures in the game. The domain relevant information includes players' positions on the board, their chips, and the number of dormants rounds already played in the game. The reliability measure for each player is r^n, as computed in Equation 1. The decisions at each round n depend only the state s^n and not on the history of past play. This violates the traditional "no-forgetting" rule that requires each decision to depend on all of the previous decisions, but is an acceptable assumption in repeated negotiation settings [6].

After players make their decisions in round n, the state s^{n+1} is updated for both players at each round to reflect the evolution of the game. This is represented by the edges from the decisions of both participants in the game at state s^n to node s^{n+1}, shown in dashed outline. In this process, the domain dependent information is updated to reflect players' positions and chips in round n. Also the reliability of the

[1]The score in the game also depends on players' positions on the board and the board layout, which we omit for expository convenience.

person in round n is aggregated using a weighted average:

$$r_H^n = (1-\alpha)\cdot r_H^{n-1} + \alpha \cdot r_H^n \quad (2)$$

where α is a decaying constant that weighs the past reliability, and is tuned empirically, as we explain in the next section. The state information s^{n+2} is updated to reflect the decisions of participants in round s^{n+1} in a similar way.

In this section we will assume the existence of the probability distribution $P(t_H^n \mid o^n, a^n, s^n)$ modeling how people transfer chips following proposal o^n and response a^n at state s^n, and the probability distribution $P(a^n \mid o^n, s^n)$ modeling how people respond to a proposal o^n in game at round n (we also assume the existence of the corresponding probability distributions for modeling people's play in round $n+1$). We detail how we learn these models in the next Section.

A key challenge to designing strategies for PAL in the game is how to assign credit to intermediate states in the game. Due to the scarcity of human data, the prediction accuracy of people's behavior decreases for later stages of the game, and constructing an influence diagram that spans the entire game is not feasible. Thus, PAL uses a heuristic value function to assign utilities to intermediate states. The value function is an estimate of the score that PAL will receive at the end of the game. This estimate is based on whether PAL gets to the goal, its score as computed by the CT scoring function described in Section 2, and game-relevant information such as the dependency relationship between players at the current state. Specifically, let s^m denote an intermediate state in the game. The node G equals true if PAL will reach the goal in the future, given that its current state is s^m. The value function is denoted $f_P(s^m, G)$ and equals PAL's score in the game given that its chip set is modified by a constant factor as follows: If PAL can get to the goal independently of the other participant, this constant is large. If PAL is dependent on the other player to get to the goal, this constant is smaller, and depends on the extent to which PAL is dependent on the other. Solving the influence diagram shown in Figure 2, s^{n+2} is chosen to be the final state and the utility node $f_P(s^{n+2}, G)$ represents the value function at that state.

Lastly, as can be seen in the influence diagram, the PAL agent is assumed to be the proposer in both rounds n and $n+1$. This significantly facilitated inference in the decision tree, because we did not need to learn a model of people's proposals. Given that there were 24 chips given to each player (see Figure 1 for an example of a board game), considering every possible proposal is infeasible. This protocol is correct for half of the game instances we collected in our setting (when PAL makes a counter proposal and is chosen to make the proposal in the next round). As we show in the Empirical section, this assumption did not impede the agent's performance.

Solving the influence diagram provides a strategy for PAL for any of its decisions given that final state is s^m. To this end, the influence diagram is converted to a decision tree and solved using backward induction. The results of this process are as follows. In the final state s^m, PAL's utility is computed using the value function $f_P(G, s^m)$ given the probability distribution $P(G \mid s^m)$ (whether PAL reaches the goal)

$$ES_P(. \mid s^m) = P(G \mid s^m)\cdot f_P(G, s^m) + P(\overline{G} \mid s^m)\cdot f_P(\overline{G}, s^m) \quad (3)$$

For any state $n < m$, we list the equations that correspond to solving the influence diagram for each of PAL's decisions. Suppose that PAL's decision is how many chips to transfer at round n after proposal o^n and response a^n. The expected score to PAL from transferring t_P^n chips is denoted $ES_P(t_P^n \mid o^n, a^n, s^n)$ and depends on its model $P(t_H^n \mid o^n, s^n)$ of people's reliability.

$$ES_P(t_P^n \mid o^n, a^n, s^n) = \sum_{t_H^n \subseteq o^n} P(t_H^n \mid o_P^n, s^n)\cdot \max_{o_P^{*,n+1}} ES_P(o_P^{*,n+1} \mid s^{n+1}) \quad (4)$$

where s^{n+1} is the updated state that realizes players' chips given that PAL transferred t_P^n chips and the human transferred t_H^n chips, and $ES_P(o_P^{*,n+1} \mid s^{n+1})$ is the score to PAL from the best proposal to make in the next state s^{n+1}.

Suppose that PAL's decision is what proposal to make at round n. The expected score to PAL from making a proposal o^n depends on its model $P(a_H^n \mid o^n, s^n)$ of how people respond to proposals. The expected utility to PAL from proposal o^n is denoted $ES_P(o_P^n \mid s^n)$ and computed as

$$ES_P(o_P^n \mid s^n) = \sum_{a_H^n \in \text{yes,no}} P(a_H^n \mid o_P^n, s^n)\cdot \max_{t_P^{*,n} \subseteq c_P^n} ES_P(t_P^{*,n} \mid o_P^n, a_H^n, s^n) \quad (5)$$

where a_H^n is the response of the person in round n, c_P^n is the set of chips in PAL's possession, and $t_P^{*,n}$ is the set of chips that PAL transfers that maximize its expected utility $ES_P(t_P^{*,n} \mid o_P^n, a_H^n, s^n)$ defined in Equation 4.

Lastly, suppose that PAL's decision is whether to accept a proposal o_H^n from the person at round n. The expected score to PAL of its response a_P^n to the proposal is denoted $ES_P(a_P^n \mid o_P^n, s^n)$ and is computed as

$$ES_P(a_P^n \mid s^n) = \max_{t_P^{*,n} \subseteq c_P^n} ES_P(t_P^{*,n} \mid o^n, a_P^{*,n}, s^n) \quad (6)$$

where $t_P^{*,n}$ is the set of chips that PAL transfers that maximize its expected utility $ES_P(t_P^{*,n} \mid o_P^n, a_P^n, s^n)$ defined in Equation 4.

4. LEARNING PEOPLE'S BEHAVIOR

In this section we describe how we constructed probabilistic models of people's behavior from data collected in the game. We defined a set of features representing aspects of the game as well as players' reliability measures. We trained classifiers for predicting people's behavior using the subset of features that performed well on a held-out set of data instances, and maximized the likelihood of the training set. These classifiers were incorporated into the influence diagram described in the last section and used by PAL to adapt to people's negotiation behavior in each country.

Past work on human-computer negotiation trained predictive models of human behavior based on their play with other people [12, 6]. There were several challenges to using this methodology in our work. First, it is logistically difficult to collect data in three countries representing different cultures under identical laboratory conditions. In particular, access to subjects in Lebanon was extremely limited. Second, the repeated nature of the game and the relatively complex rules required a session of 70-80 minutes to collect a single game instance. (We expand on the instructions given

to subjects in Section 5.) The combination of these two factors made it difficult to obtain sufficient data instances to train classifiers from each country. Third, collectivist societies such as Lebanon are more homogeneous and display less variance in the extent to which they fulfill commitments [8]. This made it difficult to predict how this population would respond to a computer player whose strategies differed from the general population.

To meet these challenges we used three sources of data to train our classifiers. First, we used the 222 game instances consisting of people playing the hand-designed agent used by Gal et al. [5]. In addition, in the U.S. and in Israel, we *were* also able to collect 112 game instances of people playing other people. Lastly, in Lebanon, we collected 64 additional games in which people played a variant of the agent used by Gal et al. that was programmed to be significantly less reliable when fulfilling its agreement. In this way, we were able to collect data of people's reactions to more diverse negotiation behavior in the game.

We defined a general set of salient features that describe people's behavior and various aspects of the game. The features are described below from the point of view of a general player i at round n in state s^n. We first describe features based on the terms defined in Section 3. The current score to i at state s^n; the Previous Reliability of i at round $n-1$, as measured by the reliability measure of Equation 1; the Aggregate Reliability at round n, as measured by the weighted average reliability of Equation 2. We further define the following additional features: the Offer Generosity of player i, which measured the difference between the number of chips offered by i and requested by i in a proposal; the Role of player i, (whether proposer or responder); the number of Dormant Rounds in which i did not move in the game. (Similar features were defined from the point of view of the other player j.)

We constructed the following probabilistic models of a general player i using the data described above. The probability $P(a_i^n \mid o^n, s^n)$ that i accepts a proposal o_j^n made by the other player j at state s^n; the probability $P(t_i^n \mid o^n, a^n, s^n)$ that player i transfers chips t_i^n after proposal o^n and response a^n at state s^n. (Note that the proposal o^n can be made by either player i or j); the probability $P(G \mid s^n)$ that i will get to the goal when it is in state s^n.

We trained multi-layered neural network classifiers to implement the various models described above using the WEKA framework.[2] We selected the features for each learning task based on their performance (measured by mean-square classification error) on a held-out set of instances as well as measuring the likelihood of the models on the training set.

The best performance for predicting people's reliability and proposal acceptance measures was obtained in Israel and in Lebanon. To explain this, we observe that the reliability of people in the data collected in Lebanon (0.67) was significantly higher than the reliability of people in the U.S. (0.289) and in Israel (0.46). This aligns with past studies showing that people care more about honor in the Middle East and are thus more reliable than in the U.S. [8].

5. EMPIRICAL METHODOLOGY

This Section describes the evaluation of PAL's performance when playing against new people in the game. To make decisions, PAL used the influence diagram described in Section 3, together with the machine learning models and the training data described in Section 4. To evaluate PAL we recruited 157 subjects from the three countries. These included 48 students studying in the Beirut area, 46 students from greater Boston area, and 63 students from universities in Israel. Each participant played a single game with the PAL agent, making a total of 157 games. At least 14 games were played in each of the dependency relationships in each country. Each participant was given an identical 30 minute tutorial on CT, consisting of a written description of the CT game, as well as an 8-minute movie that explained the rules of the game using a board that was different than those boards used in the actual study. Participants were seated in front of terminals for the duration of the study, and could not speak to each other or see their terminals. To standardize conditions with the experiments for collecting the data for learning, all participants played one game with the PAL agent, and were told they would be playing with different people.[3]

All results reported to be significant have been tested for significance in the $p < 0.05$ range using statistical ANOVA tests. We list the following three implementation details: First, the decay parameter for weighting players' reliability measures in Equation 2 was set to 0.3. This weight was tuned empirically by comparing the performance of the PAL agent on the same held-out set of instances used to evaluate the learning models in Section 4. Second, PAL chose proposals within a 10-point interval of its maximal expected score (defined in Equation 5) with uniform probability. This "trembling-hand" randomization to PAL's behavior, follows results demonstrating the benefit of randomization and unpredictability in human negotiation [3]. Third, to increase its likeness to human play, PAL did not make offers that were not present in its training set.

5.1 Comparison of Performance

Table 1 (on the following page) reports performance (in average score per game) for each of the countries and for each dependency condition. As shown by the Table, PAL was able to outperform people in all dependency conditions and in all countries: On average, PAL achieved 192.6 points in the U.S. (right-hand column in boldface), compared to 75.77 points for people; 132.6 points in Lebanon, compared to 94.86 points for people; and 152.75 points in Israel, compared to 97.85 points for people. As shown in Figure 3, PAL was also able to reach the goal significantly more often than people in all dependency conditions and in all countries.

The best performance for PAL and the worst performance for people occurred in the U.S: As Table 1 shows, PAL's average performance in the U.S. (192.6 points) was significantly higher than its performance in Lebanon (152.75 points) and Israel (132.6 points points), while people's average performance in the U.S. (75.77 points) was significantly lower than in Lebanon (94.86 points) and Israel (97.85 points). As shown in Figure 3, these results are also supported when analyzing the number of times PAL got to the goal: For all dependency conditions, PAL was able to get to the goal significantly *more* often in the U.S. than in Lebanon and

[2] `http://www.cs.waikato.ac.nz/ml/weka/`. The continuous measure of people's reliability was discretized to facilitate learning.

[3] All procedures involving people were authorized by the ethics review board of the relevant institutions.

	Co-Dependent		Independent		Dependent		Average	
	PAL	People	PAL	People	PAL	People	PAL	People
Leb.	107	73	212.18	196.3	79.3	15.3	132.6	94.86
U.S.	**188**	47	**226.25**	171.56	**163.75**	8.75	**192.6**	75.77
Isr	123	78	207.5	182.5	127.77	33.05	152.75	97.85

Table 1: Performance comparison for each condition and country

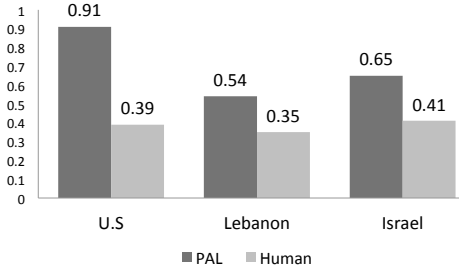


Figure 3: Getting to the Goal (in percentage of games)

Israel, and people were able to reach the goal significantly *less* often in the U.S. than in Lebanon and Israel. The rest of this Section explains how PAL was able to succeed in the various conditions given this information.

5.2 Analysis of PAL Behavior

From observing PAL's play in many games, we were able to identify several rules of behavior used by PAL that were consistent across the various dependency conditions. (1) PAL based its initial reliability measure in the games in the evaluation set (hence referred to as "evaluation games") on people's behavior in the game instances in the data used to train the models (hence referred to as "training games"). When people's reliability in the training games was consistently very low or consistently very high, it commenced the evaluation games with a low reliability measure in fulfilling its agreements. In these clear-cut situations, it makes sense for PAL to adopt a low reliability. When people's reliability measure in the training games was moderate, PAL adopted a higher reliability measure. (2) PAL was significantly less reliable when it became independent and did not depend on the other participant to reach the goal (or if it was already independent at the onset of the game). (3) PAL was significantly more reliable in games that lasted for many rounds. This caused PAL to establish reciprocal relationships with the other parties. (4) PAL was significantly less likely to fulfill agreements that allowed people to get to the goal. In the following sections, we will use these rules to analyze PAL's performance in the various dependency relationships in each country.

5.2.1 Task Co-Dependent Condition

As shown in Table 1, the highest performance of PAL in the co-dependent condition (188 points, left-hand column in boldface) occurred in the U.S. In addition, PAL got to the goal significantly more often in the U.S. (92%) than in Lebanon (35%) and in Israel (44%) in this condition. Table 2 reports the average reliability of participants in each of the conditions. As shown in the Table, the reliability of PAL was significantly lower in the U.S. (0.19) than in Israel (0.35) and in Lebanon (0.50). Also shown in the Table is that people's reliability in the U.S. (0.52) was lower than their reliability in Lebanon (0.63). To explain PAL's success in the U.S. in light of the low reliability measures exhibited by both PAL and people, we need to analyze the evolution of PAL's behavior over time.

	Co-Dependent		Independent		Dependent	
	PAL	People	PAL	People	PAL	People
Leb.	0.8	0.7	0	0.64	0.89	0.69
U.S.	0.29	0.456	0.07	0.1	0.375	0.312
Isr	0.55	0.46	0.55	0.46	0.90	0.46

Table 3: Learned and Adapted Reliability

We begin by describing PAL's reliability at the onset of the game. Table 3 shows people's average reliability in the games in the training games in each country, and PAL's adopted reliability measure after the first agreement in the games in the evaluation games. As shown by the Table, people's reliability in the training games in Lebanon (0.7) were significantly higher than their reliability in the U.S. (0.456) and in Israel (0.46). Therefore, by learned rule (1), PAL's initial reliability in Lebanon (0.8) was significantly higher than in the U.S. (0.29) or in Israel (0.55).

To explain the difference between PAL's initial reliability in the U.S. and in Israel, we need to distinguish those proposals that offered people to get to the goal, referred to as "Task Independent (TI) proposals for people". Not shown in the table, is that over 77% of the first offers made in games in the U.S. were task TI offers for people, compared to 23% of first offers made in Lebanon, and 12% of first offers made in Israel. By learned rule (3), PAL was significantly less reliable when fulfilling TI proposals for people. This is supported by Table 4, which presents the percentage of TI proposals for people in the game (out of the entire set of proposals made in the game) and the reliability of PAL when fulfilling TI offers for people. As shown by the Table, in the co-dependent condition in the U.S., the reliability of PAL when fulfilling TI offers for people (0.05, in boldface) was significantly lower than its reliability when fulfilling TI offers in Lebanon (0.14) and in Israel (0.1).

Next, we analyze PAL's behavior during the game in the co-dependent condition. We refer to those proposals that offered PAL to get to the goal as "Task Independent proposals for PAL". Table 5 shows the percentage of TI proposals for PAL in the game (out of the entire set of proposals made in the game) and the reliability of people when fulfilling TI offers for PAL. As shown in the Table the reliability of people in the U.S. when fulfilling TI offers for PAL was 0.45 (in boldface). But recall that Table 4 shows that the reliability of PAL in the U.S. when fulfilling TI offers for *people* was 0.05. This means that in the U.S. PAL was much less likely to fulfill its commitments than were people.

Further analysis revealed that the vast majority of proposals in the U.S. (93%) occurred after PAL became independent and did not need the other player to get to the goal

	Co-Dependent		Independent		Dependent		Average	
	PAL	People	PAL	People	PAL	People	PAL	People
Leb.	0.50	0.63	0.08	0.82	0.59	0.60	0.39	0.69
U.S.	**0.19**	0.52	**0.05**	0.69	**0.48**	0.62	**0.19**	0.6
Isr	0.35	0.45	0.22	0.55	0.81	0.52	0.38	0.5

Table 2: Reliability Measures for Participants

	Co-Dependent		Dependent	
	TI ratio	PAL's reliability	TI ratio	PAL's reliability
Leb.	1.05	0.14	1.53	0
U.S.	4.7	**0.05**	4.5	0.02
Isr.	3.07	0.1	4.5	0.015

Table 4: Analysis of TI offers for people

	Co-Dependent		Dependent	
	TI ratio	People's reliability	TI ratio	People's reliability
Leb.	0.94	0.4	0.73	0.37
U.S.	**1.64**	**0.45**	**1.56**	0.496
Isr.	1.7	0.25	0.72	0.54

Table 5: Analysis of TI offers for PAL

(not shown in the table). This means that PAL was able to reach the goal in early stages of the game in the U.S. According to learned rule (2), PAL was not reliable when it was independent. These findings explain how PAL was able to succeed in the U.S. while adapting a generally low reliability towards people. In contrast to the U.S., only 26% of proposals in Lebanon were made after PAL became task independent, which explains why PAL's average reliability measure in the co-dependent condition in Lebanon (0.39), shown in Table 2 was higher than its reliability in the U.S. (0.19).

To illustrate how PAL adapted its behavior in different countries the co-dependent condition, we include two examples of the evaluation games in Israel and Lebanon. In the Lebanon example, PAL began by accepting a 2-chip-for-2-chip proposal and transferring both chips following the agreement. The next agreement offered PAL the chips to get to the goal. As shown in Table 3, from the training games PAL learned that people in Lebanon were highly reliable. Therefore, PAL did not send any chips to the person following this agreement. In contrast, the person sent its promised chips to PAL, allowing PAL to get to the goal. This game was typical of Lebanon, in that games were relatively short, and people were generally reliable.

In Israel, games were longer, and people were less reliable in the training games than in Lebanon. Specifically, in our example in Israel, PAL was fully reliable following the first two agreements, while the person did not send any of its promised chips. As a result, PAL did not send any chips for the third and fourth agreements. In the fifth agreement (a 1-chip-for-1-chip proposal), PAL was fully reliable. Lastly, for the sixth agreement (a 1-chip-per-3-chip proposal), which allowed PAL to get to the goal, the human was fully reliable, while PAL did not send any of its three promised chips. This example demonstrates PAL's ability to establish a reciprocal relationship with its partner.

Lastly, we explain the difference in PAL's performance across countries. As shown in Table 5, the ratio of TI offers for PAL in the U.S. (1.64, in boldface) was almost twice as high as the ratio of TI offers for PAL in Lebanon (0.94). Thus, there were significantly more proposals that allowed PAL to reach the goal in the U.S. than in Lebanon. In contrast, the ratio of TI offers for PAL in Israel (1.7) was as high in the U.S. However, as the Table also shows, people's reliability following TI offers for PAL in Lebanon (0.4) and in the U.S. (0.45) was significantly higher than their reliability following TI offers for PAL in Israel (0.25). As a result, PAL was more likely to reach the goal in the U.S.

5.2.2 *Task Dependent Condition*

As shown by Table 1, the best performance for PAL in the task dependent condition was in the U.S. (163 points, in boldface). Table 2 shows that the lowest reliability exhibited by PAL (0.48, column "dependent"), was obtained in the U.S. In addition, PAL's reliability measure in the task dependent condition in the U.S. (0.48) was lower than that of its reliability measure in Lebanon (0.59) and Israel (0.81). These results are similar to those reported for the task co-dependent condition. However, participants' roles were not symmetric in the board games in the task dependent conditions. Specifically, in the games in which PAL was task *dependent*, people were task *independent* (and vice versa). Thus there were different factors that contributed to PAL's success in the task dependent condition.

We first observe Table 3, which shows that in the task independent condition, people's reliability in the training games in Israel (0.46) and in Lebanon (0.64) was higher than their reliability in the U.S. (0.1). Therefore, by learned rule (1), PAL commenced the evaluation games with a higher reliability in the task dependent condition in Israel (0.9) and in Lebanon (0.89) than in the U.S. (0.375).

As shown by Table 2, people's average reliability in the task independent condition in the evaluation games in the U.S. (0.69) was significantly higher than their reliability in the training games (0.1, previously shown in Table 3). As a result, PAL increased its reliability in the evaluation games in the U.S. from 0.375 (shown in Table 3, dependent column) to 0.48 (shown in Table 2, dependent column). The difference between people's reliability in the evaluation games in Lebanon and the U.S. compared to their reliability in the training games was not statistically significant. In addition, the evaluation games in Lebanon (3 rounds) were shorter than the evaluation games in Israel (6 rounds). Following learned rule (3), PAL dropped its reliability in Lebanon from 0.89 (shown in Table 3) to 0.59 (Table 2), and to a lesser extent in Israel from 0.9 to 0.81.

To illustrate PAL's strategy in the task-dependent condition, we bring an example of its play in the U.S. As shown in Table 3, from the training games PAL learned that people in the U.S. were not reliable. Therefore PAL does not send any of its chips after the first agreement (a 4-chip-per-3-chip proposal). In contrast, the person sends two of its promised 3 chips to PAL. PAL responds to this by being fully reliable

and sending all of its promised chips in the second agreement. However, the person did not send any of its promised chips to PAL in this agreement. In the third agreement (a 2-chip-per-1-chip proposal), PAL sent only one of its two promised chips to the person. The person sent PAL the chip it needs to get to the goal. As the example shows, it made sense for PAL to be partially reliable when the person is task independent at the onset of the game.

To explain PAL's success in the task dependent condition across all countries, we use Table 5, which analyzes the TI offers for PAL. As shown in the Table, the ratio of TI offers in the U.S. (1.56, in boldface) was more than twice that in Lebanon (0.73) and in Israel (0.72). This means there were significantly more offers made in the U.S. that allowed PAL to get to the goal. In addition, the reliability of people following TI offers for PAL in the task dependent condition in U.S. (0.496), shown in Table 5 was significantly higher than the reliability of PAL following TI offers for people in the task dependent condition in the U.S. (0.02), shown in Table 4. This is because when people were task dependent, PAL was task independent, and by learned rule (2), PAL was not reliable when it was independent.

5.2.3 Task Independent Condition

Recall that players in the task independent condition already possessed the necessary chips to get to the goal, and in addition could help their partners get to their own goal. As we expected, Table 1 shows that both the scores for PAL and people in this condition were higher than their scores in the task dependent and task co-dependent conditions.

Similarly to the task co-dependent condition, Table 1 shows that the highest performance by PAL and the worst performance for people were obtained in the U.S. Table 2 shows that the reliability of PAL in the task independent condition was significantly lower for each country than its reliability in the other conditions. This can be explained by rule (2), in that PAL was far less likely to fulfill agreements when it did not need its partner to get to the goal. Interestingly, the Table also shows that the reliability of people in the task independent condition was significantly *higher* than their reliability in the other conditions. To explain this discrepancy, we observe that in the games in which people were task independent, PAL was task dependent. As shown by Table 2, the reliability of PAL in the task dependent condition was significantly higher in each country than its reliability in all of the other conditions. We thus attribute people's high reliability measures when task independent to people's reciprocity to the high reliability exhibited by PAL.

6. CONCLUSIONS AND FUTURE WORK

This paper proposed a novel agent design for human-computer negotiation in different cultures. It focused on settings where participants engage in repeated rounds of negotiation and agreements are not binding. To succeed in such settings agents need to reason about the effects of their negotiation behavior over time, and to adapt to people's reaction to their behavior in different cultures. The proposed agent design combined a decision theoretic approach with classical machine learning techniques to model people's behavior. This agent was evaluated empirically by playing with 157 people in three countries—Lebanon, the U.S., and Israel. The results show that the agent was able to outperform people in all countries and when varying how parties depended on each other in the negotiations. The agent based its initial strategy on a general model of the population in each culture, and adapted its behavior to its particular partner over time. We are currently investigating the use of Markov Chain Monte Carlo sampling techniques for more efficient inference in the game.

7. ACKNOWLEDGMENTS

This work is supported in part by the following grants: Marie Curie #268362, ERC grant #267523, ARO grants W911NF0910206, W911NF1110344 and U.S. Army Research Lab and Research Office grant MURI W911NF0810144. Thanks to Louise Hindal for her help with data collection in the U.S.

8. REFERENCES

[1] A. Byde, M. Yearworth, K. Chen, C. Bartolini, and N. Vulkan. Autona: A system for automated multiple 1-1 negotiation. In *Proceedings of EC*, 2003.

[2] C. De Dreu and P. Van Lange. The impact of social value orientations on negotiator cognition and behavior. *Personality and Social Psychology Bulletin*, 21:1178–1188, 1995.

[3] R. Fisher and W. Ury. *Getting to yes*. Penguin Books New York, 1991.

[4] Y. Gal, B. Grosz, S. Kraus, A. Pfeffer, and S. Shieber. Agent decision-making in open mixed networks. *Artificial Intelligence*, 174(18):1460–1480, 2010.

[5] Y. Gal, S. Kraus, M. J. Gelfand, H. Khashan, and E. Salmon. Negotiating with people across cultures using an adaptive agent. *ACM Transactions on Intelligent Systems and Technology*, 3(1), 2012.

[6] Y. Gal and A. Pfeffer. Modeling reciprocity in human bilateral negotiation. In *AAAI'07*, 2007.

[7] M. J. Gelfand and S. Christakopoulou. Culture and negotiator cognition: Judgment accuracy and negotiation processes in individualistic and collectivistic cultures. *Organizational Behavior and Human Decision Processes*, 79(3):248–269, 1999.

[8] M. J. Gelfand, G. Shteynberg, T. Lee, J. Lun, S. Lyons, C. Bell, J. Chiao, C. Bruss, M. Al Dabbagh, Z. Aycan, et al. The cultural contagion of conflict. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 367(1589):692–703, 2012.

[9] R. A. Howard and J. E. Matheson. Influence diagrams. In *Readings on the Principles and Applications of Decision Analysis*, pages 721–762, 1984.

[10] C. 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.

[11] S. Kraus, P. Hoz-Weiss, J. Wilkenfeld, D. Andersen, and A. Pate. Resolving crises through automated bilateral negotiations. *Artificial Intelligence*, 172(1):1–18, 2008.

[12] Y. Oshrat, R. Lin, and S. Kraus. Facing the challenge of human-agent negotiations via effective general opponent modeling. In *AAMAS*, pages 377–384, 2009.

[13] A. Rosenfeld and S. Kraus. Using aspiration adaptation theory to improve learning. In *AAMAS*, 2011.

Giving Advice to People in Path Selection Problems

Amos Azaria[1] Zinovi Rabinovich[1] Sarit Kraus[1] Claudia V. Goldman[2] Omer Tsimhoni[2]

[1]Department of Computer Science Bar Ilan University Ramat Gan, Israel
[2]General Motors Advanced Technical Center Israel
{azariaa1,sarit}@cs.biu.ac.il, zr@zinovi.net, {claudia.goldman, omer.tsimhoni}@gm.com

ABSTRACT

We present a novel computational method for advice-generation in path selection problems which are difficult for people to solve. The advisor agent's interests may conflict with the interests of the people who receive the advice. Such optimization settings arise in many human-computer applications in which agents and people are self-interested but also share certain goals, such as automatic route-selection systems that also reason about environmental costs. This paper presents an agent that clusters people into one of several types, based on how their path selection behavior adheres to the paths presented to them by the agent who does not necessarily suggest their most preferred paths. It predicts the likelihood that people will deviate from these suggested paths and uses a decision theoretic approach to suggest paths to people which will maximize the agent's expected benefit, given the people's deviations. This technique was evaluated empirically in an extensive study involving hundreds of human subjects solving the path selection problem in mazes. Results showed that the agent was able to outperform alternative methods that solely considered the benefit to the agent or the person, or did not provide any advice.

1. INTRODUCTION

Research in multi-agent systems primarily encompasses systems composed of automated agents. Cooperative systems are usually described by a single utility function which all agents attempt to maximize. Competitive systems, on the other hand, may be designed and analyzed, for example, as zero sum games where the gain of one agent is the loss of another. In this paper, we focus on systems composed of both automated agents and human users. Although in general these interactive systems are cooperative, users and machines may have different interests. Each party may want to optimize different parameters, not necessarily at the expense of the other. In particular, we study automated agents interested in persuading their users to perform actions that increase the agent's utility.

Machines can try to persuade their users to perform certain actions by implementing different methods. For example, machines could provide higher rewards (e.g., score, ranking stars, etc.) when users choose actions desirable by the agents. Automated agents may disclose information not available to their users in order to encourage them to take certain actions. For example, Azaria et al. [3] have shown that agents can provide correct, although partial, information about a state of the world (unknown to the user, but relevant to his decision) and thus persuade them to take certain actions beneficial to the agent. We can also consider agents providing advice (based on the agents' advantageous information or computational power) that may lead their users to choose actions that are beneficial to the agents.

In this paper, we focus on the last method: we study how to automatically generate advice that will encourage users to choose actions preferred by the automated system.

We chose a domain, composed of human users and computers which are self-interested but also have shared goals. Consider a route selection domain where an automatic system suggests commuting routes to a human driver. Both participants in this setting share the goal of getting the driver from home to work and back. However, each participant also has its own incentives. The driver wishes to choose the route that minimizes the commuting time, while the computer may prefer taking a longer route that emits fewer pollutants, or does not pass near schools and playgrounds. The route selection domain is an example of a computationally demanding domain where even having complete knowledge is not enough for a user to solve such a problem optimally. As we will show in our experiments, finding the shortest path in large maps with many intersections may not be a trivial problem to solve. In such cases, the computer's advice might be perceived as helpful and trustful as it comes from powerful computational software.

However, the development of methods to identify agent strategies for deciding which advice to give to people is challenging. First, it is known that people are not known to maximize their monetary value. When facing noisy data, people often follow suboptimal decision strategies. This bounded rational behavior [7] is attributed to: 1) sensitivity to the context of the decision-making; 2) lack of knowledge of the user's own preferences; 3) the effects of complexity; 4) the interplay between emotion and cognition and 5) the problem of self-control. Furthermore, people discount the advice they receive from experts[5] and it was shown that if the adviser has a monetary stake in the advice being followed, people will follow its advice even less [21]. Finally, the learned model should be generalized to new environments as well as different people. To face these challenges we will integrate

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

machine learning and psychological models for predicting human response to advice.

Our study includes a 2-participant task setting for choosing a path on a large colored grid that is analogous to the route-selection problem. The person's sole incentive is to choose the shortest path, while the agent's incentives also include the number of color changes in the path. Choosing a path on the grid corresponds, for example, to selecting a route for commuting between home and work. The colors on the grid represent constraints, such as environmental and social considerations. Switching between colors on the path represents the violation of one of these constraints. The person's preferences consider the length of the route only, while the agent's preferences take into account both the length of the route as well as the number of constraint violations.

We developed the User Modeling for Path Advice (UMPA) approach for the generation of advice, comprised of a training stage and three additional steps required to learn from this data and to generate the agent's advice. We first ran experiments with human subjects to collect data on how users react when provided with advice. The system proposes three types of advice in different testing scenarios: advice that is optimal to the user, advice that is optimal to the system and advice that considers both the user's and system's preferences. We found three types of user behaviors: those that follow the system's advice, no matter how bad this advice is subjectively perceived to be; those that ignore the advice and follow their chosen path; and those that modify the advised path. This last phenomenon is very interesting since just the fact that advice is provided affects the user's choices. The user's modifications may completely change the advice or their own choice, but this change occurs only as a result of having seen such a system proposal. In particular, we noticed that users of the third type took *cuts* when solving the route selection problem. Cuts are deviations from a suggested path and are alternative segments for connecting two local points from the original path. A cut may improve the path from the user's point of view by shortening it, but may decrease the benefit to the agent.

Once we collected this data, the UMPA approach proceeded to 1) learn the percentage of types of users who will follow, ignore or modify the given advice, 2) learn with what probability each cut will be chosen for a given advised path and 3) compute the advice with the lowest expected cost for the agent given the users' predicted types and behaviors.

We evaluated the UMPA approach in an extensive empirical study comprising near 700 human subjects solving the path selection problem in four different mazes. The results showed that our UMPA agent outperformed alternative approaches for suggesting paths, based on either the user's or the system's preferences. In addition, people were satisfied with the advice provided by the UMPA agent.

2. RELATED WORK

Game theory researchers studied related research questions in the context of persuasion games. In these games, a speaker attempts to persuade a listener to accept a certain request [14, 28, 29]. Most of these works make the strong assumption that people follow equilibrium strategies. However, agents that follow equilibrium strategies when interacting with people are often not beneficial [19, 24, 3]. This can be explained by the significant experimental and other empirical evidence which indicates that people may be non-

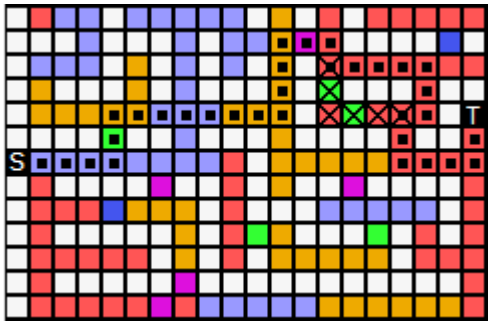


Figure 1: Path selection problem visualized in a small maze

strategic when interacting in persuasion games [13, 11, 4, 6, 8].

Route or path selection has become one of the most prominent applications of computer assisted guidance (see a survey in [17]). In fact, route guidance systems using GPS have become pervasive over the years, thanks to the significant research effort in addressing both the cognitive limitations and the range of individual preferences of human users (e.g. [12, 23]). Many of the challenges in the development of route guidance systems stem from the high variance among individuals regarding their evaluation and acceptance of route advice. This variance makes it important to tailor route advice and guidance to a specific user. To this end, a wide range of machine learning techniques are used to capture and utilize user routing preferences (e.g. [23]).

Instead of tailoring routes to users, we model user attitudes towards route advice such that the choices made by the users, after being given advice, will be beneficial to the agent. There has been some work on driver acceptance of unreliable route guidance information [15]. Antos and Pfeffer [2] designed a cooperative agent that uses graphical models to generate arguments between human decision-makers and computer agents in incomplete information settings. They use a qualitative approach that does not model the extent to which people deviate from computer-generated advice. Other works have demonstrated a human tendency to accept advice given by an adversary in games [21]. Some theoretical analysis suggests this behavior to be rational [26]. To some extent, these results were used in the framework of large population traffic manipulation (either by explicitly changing the network topology or by providing traffic information, e.g. [20, 9]). However, to the best of our knowledge, we are the first to study the combination of human choice manipulation and the personal route selection problem in a given network.

3. THE MODEL

To allow a formal discussion of the path selection problem, we employ a maze model. We assume that a user has to solve the shortest path problem within a rectangular maze either by constructing a path or by considering a path suggestion. More formally, we define a *maze* M as a grid of size $n \times m$ with one vertex marked as the source S and another vertex as the target T. Each vertex v is associated with a label $c(v)$ that we will refer to as the *color* of v. We will denote the white color or label number 0 as an *obstacle*. $x(v)$ and $y(v)$ denote the horizontal and the vertical grid coordinates

of the vertex v, respectively. We assume that the user can move along the grid edges in the four standard directions: up, down, left or right. A sequence of vertexes that does not include an obstacle and can be traversed by moving in the four standard directions is a *valid path*. In the remainder of the paper, to distinguish between vertexes of different paths, we will denote them by the path's name with a superscript: e.g. vertexes of a path π will be denoted by $\pi^1, ..., \pi^l$. A valid path will be called a *full path* if $\pi^1 = S$ and $\pi^l = T$, i.e. it begins at the source node and ends at the target node, thus solving the maze.

The *path selection problem* is modeled as the user's task to find the shortest full path through the maze. Formally, we assume that the user's cost of a path π is equal to its length, i.e. $Cost_u(\pi) = l(\pi)$. In contrast, the agent's cost depends on the length of the path and also on the number of color switching done along the path. Formally, given a color switching cost W, the agent's cost $Cost_a$ of a full path π is given by: $Cost_a(\pi) = l(\pi) + W \cdot \sum_{1 \leq i < l} \mathbf{1}\{c(\pi^i) \neq c(\pi^{i+1})\}$. We use the term *greedy path* to refer to a full path that minimizes $Cost_a$, and the term *shortest path* to refer to a full path that minimizes $Cost_u$. Notice, that there are multiple valid paths through a maze and it is possible that there are many full paths as well.

In addition to the maze grid, its color labeling and the source and target nodes, we also allow a secondary labeling of a particular full path through the maze. This labeling represents the path advised by the agent to the user. We assume that the user is aware of this labeling prior to solving the path selection problem. In fact, the advised path is part of the input to *the path selection problem*. When a user is given a maze (with or without an advised path), his goal is to solve the maze by finding the shortest full path from source S to target T. However, due to the complexity of the maze, finding such shortest path may not be trivial or clear from looking at the maze during the limited amount of time given to the user. Therefore, the user may find it beneficial to take some advice provided to him regarding which route to choose.

The *best-advised path problem* is modeled as the agent's task to compute a full path that, once presented to a user, will yield the agent the lowest expected cost.

Figure 1 visualizes the formal setting in a small maze. In the figure, obstacles are represented by the color white, while the start and the target nodes are black. In turn, the dotted nodes represent the advised path, while the crossed nodes represent a valid (partial) path selected by the user.

4. THE UMPA APPROACH

We assume the availability of training data for the prediction stages (see experiments in Section 5). UMPA is given a training set, Ψ, of tuples (M', π, μ, α) collected from experiments where people were provided with advice and where: M' is a maze; π is an advised path through the maze; α is a binary variable indicating whether the user considers π to be a good solution or not (α equals 1 or 0 respectively); and μ is the solution selected by a human user, who was presented with M' and π. In addition, we assume that Ψ includes examples (M', μ) collected from games where the agent was silent. Given a maze M (not in the maze set from the training examples), we employ a three-stage process to solve the best-advised path problem: (i) Cluster users into one of three types, depending on the extent to which their path selection behavior adheres to suggested paths that may be more beneficial to the agent than to themselves. Then we predict the likelihood that a user will belong to one of these three clusters;(ii) predicting the likelihood that people deviate from a suggested path; and (iii) generating the advised path using a decision theoretic approach which utilizes the prediction from the first two stages in order to compute the expected cost of the agent from a given path. In the next subsections we provide details of our implementation of each one of these steps.

Predicting human response to an advised path is difficult due to the diversity in people's behavior. We propose to integrate psychological models into the machine learning process. In particular, we have defined a **Seemliness-value** feature that measures the path's direction towards the target node's horizontal and vertical coordinates. This attribute will be used in the learning of UMPA. The feature value is based on the following principles known from behavioral science:

- Loss aversion [30] (Prospect theory): people dislike losing more than they like wining. Tversky and Kahneman found that losses are weighted roughly twice as much as gains. Therefore, while each step in the path toward the target contributes a single unit to the Seemliness-value, each step away from the target reduces two units from the value.

- Future discount [25]: people care more about the present than the future and therefore discount losses or gains in the future. The farther the loss or the gain is in the future, the more it is discounted. Future discounting is commonly assumed to be exponential, with some discount factor [10]. Therefore, while each step in the path toward the target at the beginning of the path adds one unit (and a step away from the target in the beginning of the path reduces two units), the contribution of any consecutive steps' is multiplied by a discount factor (which is exponential in the number of steps from the beginning of the path).

The total path Seemliness-value is calculated as a discounted sum of steps contribution along the path and is denoted $s(\phi)$. For an intuitive example, the dotted path shown in Figure 1 has a relatively high Seemliness-value since its earlier steps are in the target direction and steps in the opposite direction appear only later; however, in Figure 2 the dotted path has a relatively low Seemliness-value since the steps at the beginning of the path are in the opposite direction of the target.

4.1 Modeling Diversity in People's Reactions

Based on what was observed in the behavioral data collection experiments (as explained in Section 5), UMPA clusters users into three types: *Advice followers*, *Advice ignorers* and *Advice modifiers*. Given a new maze, when considering a path to be given as advice, UMPA would like to estimate the probability of a user belonging to one of these clusters. For this task, it first labels the examples of Ψ with one of the three types and put the examples in Ψ^l.

The labels are determined as follows. *Advice followers* are users who follow the advised path blindly without modifying it, even when believing that it is not of good quality. That is, the user of an example $(M', \pi, \mu, \alpha) \in \Psi$ is labeled as

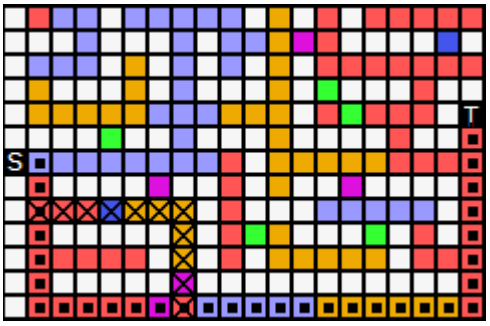


Figure 2: A second example of a path and a cut

an *Advice follower* if $\mu = \pi$ and $\alpha = 0$. Users that took the system's advice as provided and also believed that the advised path really did have good quality were included in the Advice modifiers type set (these users may have chosen the advice because it was of good quality and not because they were told to choose it).

However, most users would at least attempt to improve upon the advised path, or simply ignore it entirely. In order to characterize these users, we will introduce the concept of a *cut* and a *modified solution*.

Given two vertexes ,π^i and $\pi^{i'}$, of an advised path π, any path τ between these two vertexes (that does not otherwise intersect with π) is termed a *cut*. Although there may be an exponential number of cuts, certain human cognitive tendencies (see e.g. [12, 27]) allow us to bound the maximal cut length. All users who deviated from the advised path solely by taking cuts are termed *Advice modifiers*.

More formally, given a valid path π, we define a cut τ of length l to be a valid path such that $\exists i, \tau^1 = \pi^i$ and $\exists i' > i, \tau^l = \pi^{i'}$ and $\forall 1 < i'' < l, \nexists j, \pi^{i''} = \pi^j$. The sequence of $\pi^i, ..., \pi^{i'}$ will be called the original segment of cut τ and will be denoted by $o(\tau)$. Figure 1 and Figure 2 show examples for cuts marked by crossed nodes. We only consider cuts whose lengths are smaller than some threshold and also not much longer that their original segment. Formally, let $L_1 \in \mathbb{N}$ and $L_2 \in \mathbb{R}^+$, $l(\tau) \leq min\{L_1, L_2 \cdot l(o(\tau))\}$.

Finally, we define the *Advice ignorers* as all users who are neither *Advice modifiers* nor *Advice followers*. The relevant examples of Ψ were labeled accordingly. It is important to understand that being an advice follower does not depend on the specific maze and advice. However, deciding whether to ignore advice or use it as a baseline and modify it, depends on the specific maze and advice.

Next we compute the likelihood of users being associated with the different types as required in the first step of the UMPA approach. Based on the literature on route selection (see e.g. [18]), we presume that the proportion of *Advice modifiers* for the given advice π is strongly characterized by the overall Seemliness-value of π, denoted $s(\pi)$. In order to use the Seemliness-value of a path as an indicator for the proportion of Advice modifiers in that path, we first normalize the Seemliness-value by subtracting the average of all Seemliness-values of all paths that appear in the data-set and divide by their standard deviation. Once we have a standardized (scaleless) value, we assume that it predicts a standardized proportion of Advice modifiers in that path, therefore, this value must be unstandardized using the appropriate units found in the data-set. Formally, given Ψ^l, UMPA generates a set of tuples $\pi', s(\pi'), prop(\pi')$ where $prop(\pi')$ is the proportion of users in Ψ^l that received the advice π' and are labeled as *Advice modifiers*. Denote the average (standard deviation) of the $s(\pi')$s by $AvgSV$ ($StdSV$) and the average (standard deviation) of $prop(\pi')$s by $AvgBU$ ($StdBU$). Finally, we estimate the proportion of *Advice modifiers* to be: $p_b(\pi) = \frac{s(\pi) - AvgSV}{StdSV} \cdot StdBU + AvgBU$.

The *Advice followers* follow the advised path even if they did not evaluate it as a good path, which allows us to assume that the proportion of *Advice followers* is constant across all advised paths. We extracted this proportion from Ψ^l, and denote it by p_f. The remaining proportion of users $1 - p_f - p_b(\pi)$ is assumed to be the *Advice ignorers*. This latter set of users deviates from the advised path so much that it is possible to assume that they would have selected the same path with or without any advice given.

4.2 Predicting Advice Deviations

Given the possible advice π, UMPA estimates the probability of a user taking a specific cut τ at a given vertex π^i. We denote this probability as $p(M, \pi, \pi^i, \tau)$ and use $p(\tau)$ when the other parameters are clear from the context. UMPA assumes that the function $p(\tau)$ is a linear combination of three cut features: *cut benefit*, *cut orientation* and *cut seemliness* (see e.g. [18]).

The **Cut Benefit** measures the relative reduction in steps between the cut and the original path segment. Formally, $\frac{l(o(\tau)) - l(\tau)}{l(\tau)}$. For example, the cut shown in Figure 1 (marked with crossed nodes) has a positive benefit value since the length of the original path segment (between the first and last nodes of the cut) is greater than the length of the cut. The cut shown in Figure 2 has a benefit of 0 since the cut has the same length as the original path segment.

The **Cut Orientation** captures the tendency of human users to continue with a straight line motion. Its value depends on whether the cut or the original segment conformed to this tendency. The reference motion is the edge between the cut divergence node π^i and its predecessor in the advised path π^{i-1}. If the cut deviates from the advice by remaining in the same direction as the edge (π^{i-1}, π^i), we say that the cut has positive $+1$ orientation. If the original path segment (π^i, π^{i+1}) is similarly directed as (π^{i-1}, π^i), we say that the cut has negative -1 orientation. Otherwise, the cut's orientation is 0 (neutral). For example, in Figure 1 the value of the orientation of the cut marked by crossed nodes is 1, since the cut continues straight while the advised path turns left. The cut shown in Figure 2, however, has an orientation of -1 since the original path continues straight and the cut turns left.

The **Cut Seemliness** measures how seemly the cut is in the user's eyes. This value is calculated by subtracting the Seemliness-value of the original segment from the Seemliness-value of the cut. The seemliness of the cut shown in Figure 2 is positive since the first steps of the cut are in the same direction of the target, while the first steps in the original segment are in the opposite direction of the target.

Given that there is a very large number of cuts, it is almost impossible to collect enough examples in Ψ to learn the weights of $p(\tau)$'s features directly. Therefore, this estimation process was divided into two steps. First, UMPA estimates the probability, $r(M, \pi, \pi^i, \tau)$, that a cut τ will be taken by a user at vertex π^i, assuming that τ is the only possible cut at π^i. It was assumed that r is a linear combination

of the three cut features described above, similar to $p(\tau)$. To compute the weights of $r(\tau)$'s features, UMPA created a training set of the form $(M', \pi, \pi^i, \tau, prop(\pi^i))$, where τ is a cut of π that starts at π^i and is the cut that was taken at π^i by the highest number of users according to Ψ. $prop(\pi^i)$ is the proportion of users that visited π^i and deviated there by taking any cut. Using these examples, the weights were estimated using linear regression.

Next, $r(\tau)$ is used to compute $p(\tau)$ after normalization. For any π^i, it was assumed (based on the way that $r(\tau)$ was learned) that the probability of the deviation at π^i across all cuts is equal to the highest $r(\tau)$ value of a cut, starting at π^i. This probability is distributed across all possible cuts, starting at π^i, proportional to their $r(\tau)$ value.

4.3 Estimating the Cost of an Advised Path

Given a maze M and the possible advice π, UMPA estimates the expected cost that an agent may incur when presenting users with π. We denote this estimation by $ECost(\pi)$. This estimation is based on Ψ^l (the set of examples labeled with user types).

Notice that the contribution of the *Advice followers* is relatively easy to calculate. These are users that, independent of the maze or the particulates of the advised path π, always comply fully with π. Therefore, their contribution to $ECost(\pi)$ will always be $Cost_a(\pi)$ multiplied by the ratio of *Advice followers*.

The contribution of the *Advice ignorers* is calculated based on the data of users who received no advice. Let $\Omega_\emptyset = \{\tau | (M, \phi) \in \Psi\}$, i.e. the set of paths in Ψ selected by users who did not receive any advice. We assume that the contribution of *Advice ignorers* to *ECost* is the average agent cost on the paths in $\Omega_\emptyset$. Denote this value by $ECost_i$.

Calculating the contribution of the *Advice modifiers* to the agent's expected cost is more complex and is described hereunder. Having the estimated probability for each cut $p(\tau)$, an estimation for the agent's cost associated with *Advice modifiers* from advice π starting at π^i is denoted as $b(\pi, \pi^i)$. It can be calculated using the following recursive formulas:

$$b(\pi, \pi^{l(\pi)}) = 1$$

$$b(\pi, \pi^i) = \sum_{\tau, \tau^1 = \pi^i} p(\tau) \cdot (Cost_a(\tau) - 1) + b(\pi, \tau^{l(\tau)}) + \\ + (1 - \sum_{\tau, \tau^1 = \pi^i} p(\tau)) \cdot (b(\pi, \pi^{i+1}) + Cost_a(\pi^i \pi^{i+1}) - 1)$$

Note that the expression $Cost_a(\pi^i \pi^{i+1}) - 1$ is the agent's cost of traveling from π^i to π^{i+1}, which can either be 1 if no color switching occurs, or $W + 1$ if color switching occurs. Now, using b, UMPA can estimate the contribution of the *Advice modifiers* to the agent's expected cost of an entire path π setting $ECost_b(\pi) = b(\pi, S)$.

An efficient algorithm for computing $ECost_b$ appears in the Appendix.

Given the users' proportions as estimated in Section 4.1 and the utility contributions estimated above, we can compose the final *heuristic* estimate of the advised path cost $ECost(\pi)$, which is the expected agent's cost across all human generated path solutions in response to π:

$$ECost(\pi) = p_f \cdot Cost_a(\pi) + (1 - p_f - p_b(\pi)) \cdot ECost_i + \\ p_b(\pi) \cdot ECost_b(\pi)$$

4.4 Searching for Good Advice

Searching for advice is done by transforming the maze(grid) to a tree such that the start node, S, is associated with the root of the tree. Each node in the tree is associated with a vertex in the maze. A node n_v in the tree that is associated with the vertex v will have an offspring which is associated with v' if no ancestor of n_v is associated with v' and v' is connected to v in the grid. Note that a vertex in the grid might be associated with many nodes in the tree. When given a node n_v in the tree that is associated with the vertex v, there is a unique path in the tree from the root node of the tree to n_v that is associated with a path on the grid from S to v. We denote this path as θ.

A^* [16], which is a best-first search algorithm in graphs, uses the sum of a cost function and a heuristic function in order to determine which node to view next. We use the A^* search algorithm on the tree, to find a path π from the root node S to any target T. The cost function for a given node n_v is $ECost(\theta)$ and the agent uses the minimal agent cost of traveling between v and T as the heuristic function of n_v in the tree. We use Dijkstra's algorithm, which is an efficient algorithm for calculating the shortest path from a given node to all other nodes in a graph, starting at T, in order to calculate the minimal agent cost to travel from each vertex to T.

To limit the manipulation effect of UMPA, the search only considers paths with cuts where the agent does not gain by the user taking them. That is, the agent prefers that the user takes the advised path and does not benefit from his deviation. Formally, UMPA only considers paths such that, for any suffix $\sigma = \pi^i \cdots \pi^{l(\pi)}, i \geq 1$, $ECost(\sigma) \geq Cost_a(\sigma)$ holds. If A^* stops with a path that does not satisfy the condition above it will be rejected, and A^* will be forced to continue the search.

5. EXPERIMENTAL EVALUATION

We have developed an online system that allows people to solve path selection problems in a maze. It can be accessed via `http://azariaa.com/selfmazeplayer.swf`. The maze design was chosen to remove all effects of familiarity with the navigation network from the experiments. Furthermore, every human subject was presented with a single instance of the problem in order to exclude effects of learning or trust. We ran two kinds of experiments. First, the experiments were aimed at collecting data on users' behaviors when facing advice that either benefited the users or the system utilities regarding route selection. Second, after the UMPA approach was applied using the collected data, we ran experiments to validate our hypothesis regarding users' behavior change as a result of providing them with advice adapted to the user's behavior as learned in the first experiments. Furthermore, the main goal has been to test the hypothesis that UMPA outperformed all of the other advice generator methods that we considered.

Participation in our study consisted of 681 subjects from the USA: 383 females and 298 males. The subjects' ages ranged from 18 to 72, with a mean of 37.

5.1 Methodology

5.1.1 Running Experiments on Amazon Mechanical Turk

All of our experiments were run using Amazon Mechanical Turk (AMT) [1], a crowd sourcing web service that coordinates the supply and demand of tasks which require human intelligence to complete. Amazon Mechanical Turk has become an important tool for running experiments with human subjects and was established as a viable method for data collection [22]. We took several actions to encourage subjects to truly attempt to find the shortest path: we only selected workers with a good reputation; a set of questions, designed to verify understanding of the task, was presented to the subjects prior to the task execution; and as a stimulus, all subjects were guaranteed a monetary bonus inversely proportionate to the length of the path that they selected. Our previous experience in running experiments on Mechanical Turk demonstrated that almost all subjects have considered our tasks seriously. We asked a group of university students and Mechanical Turk workers to perform the same task and found that the average score of the Amazon Turk workers was higher than that of the students. Thus, our own experience confirms other studies [22] about the viability of this medium for empirical research.

5.1.2 Experimental Setup

Each experiment consisted of a colored-maze panel similar to the one depicted in Figure 1. A single panel was shown to each participant. The user's task was to select the shortest path through the maze that connected the source and target nodes. When subjects were presented with advice from the system, they *were informed* that this advice was calculated to reduce the number of color switches in addition to minimizing the path length. We implicitly asked the subjects a question regarding the system's intention to make sure that they understood this crucial point. We used four distinct mazes, all of size 80×40. These mazes were complex enough so that users would find it difficult to compute the shortest path in the limited time allotted for the task. We set the weight W for color switching to 15.

We ran four training sessions to learn user behaviors from three mazes. Then we ran our UMPA algorithm on the fourth maze to compute the advice, using information about this maze and the parameters learned from the other three mazes (we did this for each one of the four mazes). That is, UMPA's results are averaged over four different mazes and training and testing data were strictly separated.

Finally, we presented the subjects with post-task questions that were designed to assess the general attitude towards computer advice and the subjective evaluation of the advised path quality.

5.1.3 Basic Algorithms

We compared the performance of our UMPA algorithm to the following three cases:

- *No advice (silent)* – no advice is presented on the maze panel,
- *Shortest path* – the advice presented corresponds to the shortest path from source to target,
- *Greedy* – the advice that the user gets is the path computed to minimize the agent's cost of traversing it, $Cost_a$.

The *Shortest* solution is the one that minimizes the cost of the user and, therefore, we expect that its acceptance by the users will be high. Moreover, the number of *advice ignorers* will be small and the probability of deviation will be low as well. However, since the agent's cost for this path is usually high, we expect that presenting *Shortest* will yield the agent a relatively high average cost. When providing *Greedy* advice, we run the risk that most of the users will ignore it, while the ones that will accept it will yield the highest benefits to the agent. We first compared the agent's average cost when providing any one of these three types of advice. (This comparison was performed using ANOVA, a method of analysis used to determine the level of statistical significance when dealing with more than two groups). Then we chose the one that was best for the agent and compared the UMPA solution to this *baseline algorithm*. Then we considered UMPA estimation methods, its performance vs. the baseline algorithm and whether it decreased the user's benefit and satisfaction or if it was mutually beneficial for both the agent and the user.

5.2 Basic Results

We calculated the effects that Silent, Shortest and Greedy types of advice have on the average agent cost across paths selected by users in our experiments. The corresponding three bar charts on the left of Figure 3 summarize the results (the lower the better). The average costs over four mazes of types Silent, Shortest and Greedy were 559.73, 559.55 and 501.68, respectively. That is, the paths chosen by users after receiving *Greedy* advice have resulted in a significantly ($p < 0.001$) lower cost to the agent than the cost attained when the other two types of advice were given (Shortest and Silent).

We have also studied the statistics of the advice effect on the user's cost (see three left-most bar charts of Figure 4). As expected, the cost of the paths chosen by users was significantly lower (130.85) when *Shortest* advice was provided, than when the other two types of advice were given (Greedy (144.6) and Silent (142.75)). Moreover, we wanted to check whether giving advice that results in the lowest costs to the agent can also decrease the costs to the users, when compared to the case where no advice is provided. The results were mixed and no significant difference was found between Greedy and Silent. That is, while *Greedy* advice significantly decreased the agent's cost, it did not significantly increase the user's costs. We concluded that the UMPA advice generation algorithm should be compared to the case where *Greedy* advice is provided.

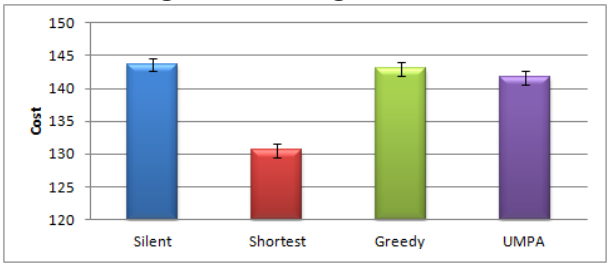
5.3 UMPA Advice Algorithm Performance

We set the UMPA parameters as follows: the length of a cut L_1 was bound to 40; a cut's potential increase in length L_2 to 20% of the corresponding original segment and the discount factor δ in the cut-seemliness feature calculation was set to 0.95. These parameters where chosen to optimize prediction accuracy within computational limitations.

The first step in the evaluation of our UMPA algorithm was to verify the effectiveness in computing $p(M, \pi, \pi^i, \tau)$ (i.e., the *predicted* number of users that will take cut τ when facing divergence node i, when advice π was provided in maze M). We found a high correlation (0.77) between this prediction and the actual fraction of users who took it when reaching the cut's divergence node. A high correlation (0.7) was also found between the actual fraction of users that took advice π or manipulated it, *the Advice modifiers* and our

Figure 3: Average agent's costs

Figure 4: Average users' costs



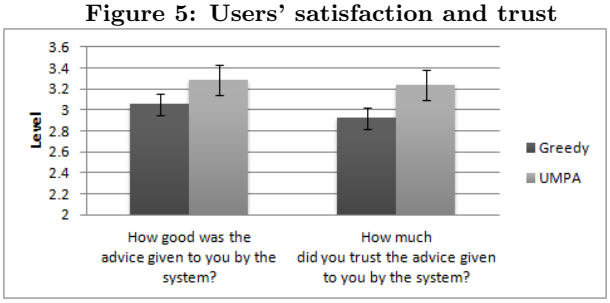
predicted number of such users, $p_b(\pi)$. Finally, we obtained a high correlation (0.76) between the estimated value of advice π, $ECost_a(\pi)$ and the empirical average value of the actual paths selected in response to advice π. This is significant since the correlation between the agent's cost of π itself and the empirical average of the selected path was only 0.06.

We then compared the average cost attained by the agents when users chose paths after receiving either the UMPA-based advice or *Greedy* advice. Consider the two corresponding bar charts on the right side of Figure 3 (the lower the better). UMPA's average costs over the four mazes was 484.95 compared with the Greedy advice that was 501.68. That is, on average, the UMPA approach outperformed *Greedy* advice, resulting in significantly lower costs ($p < 0.05$) for the agent.

We also compared the average cost incurred by the paths chosen by users to the users themselves when receiving the advice provided by the UMPA algorithm and *Greedy* advice (see the two right-most bar charts of Figure 4). To our surprise, the average results attained by the users that were given the UMPA advice (142.33) were significantly better (lower cost) than those attained by users who were presented with *Greedy* advice (142.33) ($p < 0.05$). In summary, when comparing the results obtained by running two advice generation techniques (one provides UMPA advice and the other provides *Greedy* advice), we conclude that UMPA-based advice outperforms *Greedy* advice. That is, the average cost incurred by the agent when users selected their paths and the average cost incurred by the human users would decrease significantly when the users were provided with UMPA advice. So UMPA manipulative advice is indeed *mutually* beneficial when compared with *Greedy* advice.

Finally, we considered the subjective view of the users on the paths that were advised. Users were presented with the following questions after they finished the route selection task: (i) "How good was the advice given to you by the system?" and (ii) "How much did you trust the advice given to you by the system?" The possible answers were on a scale

Figure 5: Users' satisfaction and trust



of 1-5, where 5 indicates the highest satisfaction and 1 the lowest satisfaction. The results are presented in Figure 5. Regarding the first question, UMPA advice was considered to be significantly better than *Greedy* advice, with $p < 0.05$. The average rating for UMPA was 3.29 and the average rating for *Greedy* was only 3.05. Similarly, with respect to trust, the average rating of UMPA was 3.23 whereas the average rating of *Greedy* was only 2.92, i.e., users trusted UMPA advice significantly more than *Greedy* advice ($p < 0.05$).

6. CONCLUSIONS AND FUTURE WORK

This paper presents an innovative computational model for advice generation in human-computer settings where agents are essentially self-interested but share some common goals with the human. To assess the potential effectiveness of our approach, we performed an extensive set of path selection experiments in mazes. Results showed that the agent was able to outperform alternative methods that either solely considered the agent's or the person's benefit, or did not provide any advice.

The approach that was described in this paper can be technically summarized as follows: first, sample user response to basic advice patterns. Then create a model of the response using machine learning and relevant psychological models. Finally, solve inverse kinematics of the model in order to find the most profitable advice. This technical structure can be repeated in any domain or task where a self-interested agent can provide advice to a human user and the basic response data can be obtained. Specifically, whenever the task can be converted into a path-in-graph formulation (e.g. supply-chain plans), our solution can become an out-of-the box (yet tunable) method for advice provision.

Given these encouraging results, we expect that the proposed technology can be applied to other applications where the agent's goal is to provide people with advice that will lead them to take beneficial actions. Recent applications, such as coaching humans in weight-loss programs, programs to help quit smoking or online service providers such as automated travel agents are domains that are promising.

In future work, we will extend this approach to settings in which people and computers interact repeatedly, requiring the agent to reason about the effects of its current advice on people's future behavior.

7. ACKNOWLEDGMENTS

We thank Ya'akov Gal, Shira Abuhatzera and Ariella Richardson for their helpful comments and acknowledge ERC (grant #267523) for supporting this research.

8. REFERENCES

[1] Amazon. Mechanical Turk services. http://www.mturk.com/, 2010.

[2] D. Antos and A. Pfeffer. Using reasoning patterns to help humans solve complex games. In *IJCAI*, pages 33–39, 2009.

[3] A. Azaria, Z. Rabinovich, S. Kraus, and C. Goldman. Strategic information disclosure to people with multiple alternatives. In *Proc. of AAAI*, 2011.

[4] A. Blume, D. V. DeJong, Y.-G. Kim, and G. B. Sprinkle. Evolution of communication with partial common interest. *Games and Economic Behavior*, 37(1):79 – 120, 2001.

[5] S. Bonaccio and R. S. Dalal. Advice taking and decision-making: An integrative literature review and implications for the organizational sciences. *Organizational Behavior and Human Decision Processes*, 101(2):127–151, 2006.

[6] H. Cai and J. T.-Y. Wang. Overcommunication in strategic information transmission games. *Games and Economic Behavior*, Vol. 56, Issue 1:7–36, July 2006.

[7] C. F. Camerer. *Behavioral Game Theory. Experiments in Strategic Interaction*, chapter 2. Princeton University Press, 2003.

[8] Y. Chen. Perturbed communication games with honest senders and naive receivers. *Journal of Economic Theory*, 146(2):401 – 424, 2011.

[9] C. G. Chorus, E. J. Molin, and B. van Wee. Travel information as an instrument to change car drivers travel choices. *EJTIR*, 6(4):335–364, 2006.

[10] L. de Alfaro, T. Henzinger, and R. Majumdar. Discounting the future in systems theory. In J. Baeten, J. Lenstra, J. Parrow, and G. Woeginger, editors, *Automata, Languages and Programming*, volume 2719 of *Lecture Notes in Computer Science*, pages 192–192. Springer Berlin / Heidelberg, 2003.

[11] J. W. Dickhaut, K. A. McCabe, and A. Mukherji. An experimental study of strategic information transmission. *Economic Theory*, 6(3):389–403, November 1995.

[12] M. Duckham and L. Kulik. "Simplest" paths: Automated route selection for navigation. In *LNCS*, volume 2825, pages 169–185, 2003.

[13] R. Forsythe, R. Lundholm, and T. Rietz. Cheap talk, fraud, and adverse selection in financial markets: Some experimental evidence. *The Review of Financial Studies*, Vol. 12, No, 3:481–518, Fall 1999.

[14] J. Glazer and A. Rubinstein. On optimal rules of persuasion. *Econometrica*, 72(6):1715–1736, 2004.

[15] R. J. Hanowski, S. C. Kantowitz, and B. H. Kantowitz. Driver acceptance of unreliable route guidance information. In *Proc. of the Human Factors and Ergonomics Society*, pages 1062–1066, 1994.

[16] 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*, 4(2):100–107, Feb. 1968.

[17] M. Hipp, F. Schaub, F. Kargl, and M. Weber. Interaction weaknesses of personal navigation devices. In *Proc of AutomotiveUI*, 2010.

[18] H. H. Hochmair and V. Karlsson. Investigation of preference between the least-angle strategy and the initial segment strategy for route selection in unknown environments. In *Spatial Cognition IV*, volume 3343 of *LNCS*, pages 79–97, 2005.

[19] P. Hoz-Weiss, S. Kraus, J. Wilkenfeld, D. R. Andersend, and A. Pate. Resolving crises through automated bilateral negotiations. *Artificial Intelligence journal*, 172(1):1–18, 2008.

[20] S. K. Hui, P. S. Fader, and E. T. Bradlow. Path data in marketing. *Marketing Science*, 28(2):320–335, 2009.

[21] X. J. Kuang, R. A. Weber, and J. Dana. How effective is advice from interested parties? *J. of Economic Behavior and Organization*, 62(4):591–604, 2007.

[22] G. Paolacci, J. Chandler, and P. G. Ipeirotis. Running experiments on Amazon Mechanical Turk. *Judgment and Decision Making*, 5(5), 2010.

[23] K. Park, M. Bell, I. Kaparias, and K. Bogenberger. Learning user preferences of route choice behaviour for adaptive route guidance. *IET Intelligent Transport Systems*, 1(2):159–166, 2007.

[24] N. Peled, Y. Gal, and S. Kraus. A study of computational and human strategies in revelation games. In *Proc. of AAMAS*, 2011.

[25] H. Rachlin and L. Green. Commitment, choice and self-control. *J. Exp. Anal. Behav.*, 17:15–22, 1972.

[26] L. Rayo and I. Segal. Optimal information disclosure. *Journal of Political Economy*, 118(5):949–987, 2010.

[27] K.-F. Richter and M. Duckham. Simplest instructions: Finding easy-to-describe routes for navigation. In *Proc. of the Geographic Information Sience*, volume 5266 of *LNCS*, pages 274–289, 2008.

[28] I. Sher. Credibility and determinism in a game of persuasion. *Games and Economic Behavior*, 71(2):409 – 419, 2011.

[29] J. Sobel. Giving and receiving advice. Working Paper, August 2010.

[30] A. Tversky and D. Kahneman. Loss Aversion in Riskless Choice: A Reference-Dependent Model. *The Quarterly J. of Economics*, 106(4):1039–1061, 1991.

APPENDIX

Input: A maze, with an advised path π.

Output: $ECost_b(\pi)$ – estimated cost contributed by Advice modifiers

1: $ECost_b \leftarrow Cost_a(\pi)$.
2: $vec \in \mathbf{R}^{l(\pi)} \leftarrow \vec{0}$. $vec(0) = 1$.
3: **for** each $i < l(\pi)$ **do**
4: **for** each cut τ s.t. $\tau^1 = \pi^i$ **do**
5: {**Predict the fraction of Advice modifiers who take the cut**}
$a(\tau) \leftarrow (1 + \sum_{j<i} vec[j]) \cdot p(\tau)$
6: $ECost_b \leftarrow ECost_b + (Cost_a(\tau) - Cost_a(o(\tau))) \cdot a(\tau)$.
7: {**Update mass at cut entry point.**}
$vec[i] \leftarrow vec[i] - a(\tau)$
8: {**Update the cut exit point**}
$vec[j|\pi^j = \tau^{l(\tau)}] \leftarrow vec[j] + a(\tau)$
9: **return** $ECost_b$.

Intuitively, the algorithm's basic assumption is that the set of users forms a continuous unit mass. The algorithm then traces the flow of this unit of mass along different cuts that diverge (or converge) at vertexes along the advised path.

This algorithm can be implemented with a complexity of $O(\#cuts + l(\pi))$.

Combining Human and Machine Intelligence in Large-scale Crowdsourcing

Ece Kamar
Microsoft Research
Redmond, WA 98052
eckamar@microsoft.com

Severin Hacker*
Carnegie Mellon University
Pittsburgh, PA 15289
shacker@cs.cmu.edu

Eric Horvitz
Microsoft Research
Redmond, WA 98052
horvitz@microsoft.com

ABSTRACT

We show how machine learning and inference can be harnessed to leverage the complementary strengths of humans and computational agents to solve crowdsourcing tasks. We construct a set of Bayesian predictive models from data and describe how the models operate within an overall crowdsourcing architecture that combines the efforts of people and machine vision on the task of classifying celestial bodies defined within a citizens' science project named Galaxy Zoo. We show how learned probabilistic models can be used to fuse human and machine contributions and to predict the behaviors of workers. We employ multiple inferences in concert to guide decisions on hiring and routing workers to tasks so as to maximize the efficiency of large-scale crowdsourcing processes based on expected utility.

Categories and Subject Descriptors

I.2 [**Distributed Artificial Intelligence**]: Intelligent agents

General Terms

Design, Algorithms, Economics

Keywords

crowdsourcing, consensus tasks, complementary computing, decision-theoretic reasoning

1. INTRODUCTION

Efforts in the nascent field of human computation have explored methods for gaining programmatic access to people for solving tasks that computers cannot easily perform without human assistance. Human computation projects include work on crowdsourcing, where sets of people jointly contribute to the solution of problems. Crowdsourcing has been applied to solve tasks such as image labeling, product categorization, and handwriting recognition. To date, computers have been employed largely in the role of platforms for recruiting and reimbursing human workers; the burden of managing crowdsourcing tasks and making hiring decisions has relied on manual designs and controls. However, interest has been growing in applications of learning and planning to crowdsourcing.

We investigate principles and algorithms for constructing crowdsourcing systems in which computer agents learn about tasks and about the competencies of workers contributing to solving the tasks, and make effective decisions for guiding and fusing multiple contributions. As part of this investigation, we demonstrate how we can leverage the complementary strengths of humans and computer agents to solve crowdsourcing tasks more efficiently. We describe the operation of key components and overall architecture of a methodology we refer to as CrowdSynth, and demonstrate the operation and value of the methods with data and workload drawn from a large-scale legacy crowdsourcing system for citizen science.

We focus on solving *consensus tasks*, a large class of crowdsourcing. With consensus tasks the goal is to identify a hidden state of the world by collecting multiple assessments from human workers. Examples of consensus tasks include *games with a purpose* (e.g., image labeling in the ESP game) [13], paid crowdsourcing systems (e.g., product categorization in Mechanical Turk) [6], and citizen science projects (e.g., efforts to classify birds or celestial objects). Consensus efforts can be subtasks of larger tasks. For example, a system for providing real-time traffic flow and predictions may contact drivers within targeted regions for reports on traffic conditions [8].

We describe a general system that combines machine learning and decision-theoretic planning to guide the allocation of human effort in consensus tasks. Our work derives from a collaboration with the Galaxy Zoo citizen science effort [1], which serves as a rich domain and source of data for evaluating machine learning and planning methods as well as for studying the overall operation of an architecture for crowdsourcing. The Galaxy Zoo effort was organized to seek help from volunteer citizen scientists on the classification of thousands of galaxies that were previously captured in an automated astronomical survey, known as the Sloan Digital Sky Survey (SDSS). The project has sought assessments via the collection of multiple votes from non-experts. Beyond votes, we have access to a database of SDSS image analysis data, containing 453 image features for each galaxy, which were extracted automatically via automated machine vision.

We shall describe how successful optimization of the engagement of people with Galaxy Zoo tasks hinges on models learned from data that have the ability to predict the ul-

*Severin Hacker contributed to this research during an internship at Microsoft Research.

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

timate classification of a celestial objects, including objects that are undecidable, and of the next votes that will be made by workers. Such predictions enable the system to balance the expected benefit versus the costs of hiring a worker. We formalize Galaxy Zoo as a Markov Decision Process with partial observability, using likelihoods of outcomes generated by the predictive models for states of ground truth and for worker assessments. We demonstrate that exact computation of the expected value of hiring workers on tasks is infeasible because of the long horizon of Galaxy Zoo tasks. We present approximation algorithms and show their effectiveness for guiding hiring decisions. We evaluate the methods by drawing votes from the dataset collected from the Galaxy Zoo system during its operation in the open world. The evaluations show that the methods can achieve the maximum accuracy by hiring only 47% of workers who voted in the open-world run of the system. The evaluations call attention to the robustness of different algorithms to uncertainties in the inferences from the learned predictive models, highlighting key challenges that arise in fielding large-scale crowdsourcing systems.

2. RELATED WORK

Modeling workers and tasks has been an active area of crowdsourcing research. Whitehill et al. apply unsupervised learning to simultaneously predict the correct answer of a task, the difficulty of the task and the accuracy of workers based on some assumptions about the underlying relationships between the answer, the task, and workers [14]. Dai et. el. assume that worker reports are independent given the difficulty of tasks, and learn models of workers and task quality under this independence assumption [3].

Previous work on decision-theoretic reasoning for crowdsourcing tasks focused on tasks that can be decomposed into smaller tasks [10], and on workflows that are composed of an improve and verify step, which can be solved via methods that perform short lookaheads [3]. In a related line of work, researchers proposed greedy and heuristic approaches for active learning in crowdsourcing systems [11]. Our work differs from previous approaches in the generality of the Bayesian and decision-theoretic modeling, and in our focus on learning and executing expressive models of tasks and workers learned from real-world data.

3. SOLVING CONSENSUS TASKS

A task is classified as a *consensus task* if it centers on identifying a correct answer that is not known to the task owner and there exists a population of workers that can make predictions about the correct answer. Formally, let t be a consensus task and A be the set of possible answers for t. There exists a mapping $t \rightarrow \bar{a} \in A$ that assigns each task to a correct answer.

Figure 1 presents a schematic of components and flow of analysis of a utility-directed *consensus system*. The consensus system takes as input a consensus task. The system has access to a population of workers, who are able to report their noisy inferences about the correct answer. Given that $L \subseteq A$ is a subset of answers that the system and workers are aware of, a report of a worker includes the worker's vote, $v \in L$, which is the worker's prediction of the correct answer. The system can hire a worker at any time or may decide to terminate the task with a prediction about the correct answer of the task based on reports collected so far ($\hat{a}$). The goal of the system is to predict the correct answer of a given task based on potentially noisy worker reports while considering the cost of resources.

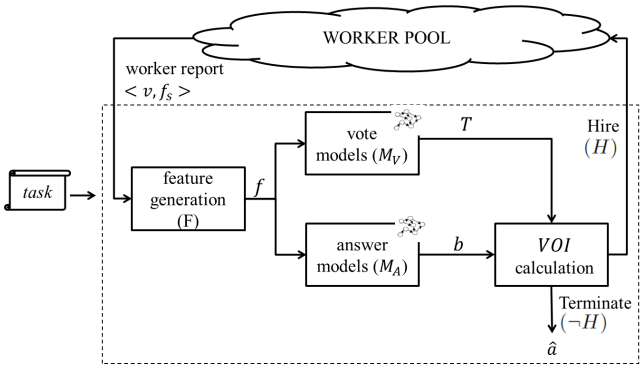


Figure 1: CrowdSynth: Key components and flow of analysis.

A successful system for solving consensus tasks needs to manage the tradeoff between making more accurate predictions about the correct answer by hiring more workers, and the time and monetary costs for hiring. We explore the opportunity to optimize parameters of this tradeoff by making use of a set of predictive models and a decision-theoretic planner.

The modeling component is responsible for constructing and using two groups of predictive models: *answer models* for predicting the correct answer of a given consensus task at any point during the process of acquiring votes, and *vote models* that predict the next state of the system by predicting the votes that the system would receive from additional workers should they be hired, based on the current information state. The modeling component monitors the worker population and task execution, and collects data about task properties and worker statistics, votes collected, and feedback received about the correct answer. A case library of execution data is used to build the answer and vote models.

We construct the answer and vote prediction models with supervised learning. Log data of any system for solving consensus tasks provides labeled examples of workers' votes for tasks. Labeled examples for training answer models are obtained from experts who identify the correct answer of a task with high accuracy. When expert opinion is not available, the consensus system may assume that the answer deduced from the reports of infinitely many workers according to a predetermined consensus rule is the correct answer of a given task (e.g., the majority opinion of infinitely many workers). To train answer models without experts, the system collects many worker reports for each task in the training set, deduces the correct answer for each task, and records either the consensus answer or the undecidable label.

Both answer and vote models are inputs to the planner. Vote models constitute the stochastic transition functions used in planning for predicting the future states of the model. The planner makes use of answer models for estimating the confidence on the prediction so that the planning component can decide whether to hire an additional worker.

The decision-theoretic planner models a consensus task as

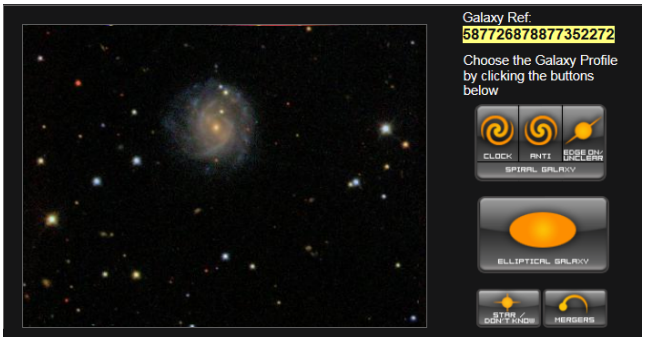


Figure 2: Galaxy Zoo interface for acquiring votes.

a Markov Decision Process (MDP) with partial observability. The MDP model is able to represent both the system's uncertainty about the correct answer and uncertainty about the next vote that would be received from workers. The planner computes the expected value of information (VOI) that would come with the hiring of an additional worker and determines whether the system should continue hiring (H) or terminate ($\neg H$) at any given state to maximize the total utility of the system. The utility is a combination of the reward (or punishment) of the system for making a correct (or incorrect) prediction and cost for hiring a worker.

3.1 Tagging Galaxies as a Consensus Task

In Galaxy Zoo, volunteers provide votes about the correct classifications of millions of galaxies that have been recorded in an automated sky survey [1]. Crowdsourcing provides a novel way for astronomers to reach a large group of workers around the world and collect millions of classifications under the assumption that the consensus of many workers provide the correct classification of a galaxy.

Figure 2 displays the main interface between the system and workers for collecting worker reports. The system displays images of celestial objects taken from SDSS and asks workers to classify them into 6 possible classes: elliptical galaxy, clockwise spiral galaxy, anticlockwise spiral galaxy, other spiral galaxy, and stars and mergers.

The dataset collected to date includes over 34 million worker reports obtained for 886 thousand unique galaxies. We use a subset of this dataset to train and test predictive models. We use another subset to simulate the real-time execution of the methodology within a prototype system named CrowdSynth and evaluate its performance under varying domain conditions.

4. PREDICTIVE MODELS FOR CONSENSUS TASKS

We now focus on the construction of predictive models for answers and votes.

4.1 Datasets

We shall perform supervised learning from a case library that includes log entries collected during the operation of the Galaxy Zoo system. Each log entry corresponds to a worker report collected for a galaxy. A worker report is a combination of a vote ($v_i \in L$), and information and statistics (f_{s_i}) about the worker delivered v_i. v_i represents a worker's prediction of the correct answer (e.g., elliptical) and f_{s_i} includes the worker's identity, the dwell time of the worker, the time and day the report is received. In addition to v_i and f_{s_i}, a log entry for a galaxy includes the visual features of a galaxy (SDSS features). We divided the celestial objects in the Galaxy Zoo dataset into a training set, a validation set and a testing dataset, each having 628354, 75005, and 112887 galaxies respectively.

We defined sets of features that summarize task characteristics, the votes collected for a task, and the characteristics of the workers reported for the task. f, the set of features for a galaxy, is composed of four main sets of features: f_0, *task features*, f_v, *vote features*, f_w, worker features, and f_{v-w}, *vote-worker features*. *Task features* include 453 features that are extracted automatically from sky survey images by making use of machine vision [9]. These features are available for each galaxy in the system in advance of votes from workers. *Vote features* capture statistics about the votes collected by the system at different points in the completion of tasks. These features include the number of votes collected, the number and ratio of votes for each class in L, the entropy of the vote distribution, and the majority class. *Worker features* include attributes that represent multiple aspects of the current and past performance, behaviors, and experience of workers contributing to the current task. These features include the average dwell time of workers on previous tasks, average dwell time for the current task, their difference, mean and variance of number of tasks completed in past, and average worker accuracy on aligning with the correct answer. We use the training set to calculate features about a worker's past performance. Finally, *vote-worker features* consist of statistics that combine vote distributions with worker statistics. These include such attributes as the vote by the most experienced worker, the number of tasks responded by a worker, the vote of the worker who has been most correct, and her accuracy.

A feature extraction function F takes a galaxy task and a history of worker reports $h_t = \{< v_1, f_{s_1} >, ..., < v_t, f_{s_t} >\}$, and creates f, the set of features described here, as input to the predictive models.

Based on an analysis on the dataset, the designers of the Galaxy Zoo system identified the following consensus rule: After hiring as many workers as possible for a celestial object (minimum 10 reports), if at least 80% of the workers agree on a classification (e.g., elliptical, spiral, etc.), that classification is assigned to the celestial object as the correct answer. Experts on galaxy classifications note that the correct answers assigned to galaxies with this rule agree with expert opinions in more than 90% of the cases, and thus using this rule to assign ground truth classification to galaxies does not significantly diminish the quality of the system [9]. In our experiments, we consider galaxies with at least 30 votes and apply this rule to generate labeled examples.

Not all galaxies in the dataset have votes with 80% agreement on a classification when all votes for that galaxy are collected. We classify such galaxies as "undecidable" and we define $A = L \cup \{undecided\}$, where L is the set of galaxy classes. Having undecidable galaxies introduces the additional challenge for the predictive models of identifying tasks that are undecidable, so that the system does not spend valuable resources on tasks that will not converge to a classification. M_A, the answer model, is responsible for deciding if a galaxy is decidable as well as identifying the correct galaxy

class if the galaxy is decidable without knowing the consensus rule that is used to assign correct answers to galaxies. Because the number of votes each galaxy has in the dataset varies significantly (minimum 30, maximum 95, average 44), predicting the correct answer of a galaxy at any step of the process (without knowing how many votes the galaxy has eventually) is a challenging prediction task. For example, two galaxies with the same vote distribution after 30 votes may have different correct answers.

We perform Bayesian structure learning to data from the case library to build probabilistic models that make predictions about consensus tasks. For any given learning problem, the learning algorithm selects the best predictive model by performing heuristic search over feasible probabilistic dependency models guided by a Bayesian scoring rule [2]. We employ a variant learning procedure that generates decision trees for making predictions.

4.2 Predicting Correct Classification

We now explore the challenge of predicting the correct answer of a consensus task based on noisy worker reports. We first implement several basic approaches as proposed by previous work [12], and then present more sophisticated approaches that can better represent the dependency relationships among different features of a task.

The most commonly used approach in crowdsourcing research for predicting the correct answer of a consensus task is majority voting. This approach does not perform well in the galaxy classification domain because it incorrectly classifies many galaxies, particularly the tasks that are undecidable.

Next, we implement two approaches that predict the correct answer using Bayes' rule based on the predictions of the following models: $M_A(\bar{a}, F(f_0, \emptyset))$, a prior model for the correct answer, and $M_{V'}(v_i, \bar{a}, F(f_0, h_{i-1}))$, a vote model that predicts the next vote for a task conditional on the complete feature set and the correct answer of the galaxy. Because v_i is the most informative piece of a worker's report and predicting f_{s_i} is difficult, we only use $M_{V'}$ model to predict a worker's report.

The Naive Bayes approach makes the strict independence assumption that worker reports are independent of each other given task features and the correct answer of the task. Formally, $Pr(\bar{a}|f)$, the likelihood of the correct answer being $\bar{a}$ given feature set f, is computed as below:

$$Pr(\bar{a}|f) = Pr(\bar{a}|F(f_0, h_t)) \approx M_A(\bar{a}, F(f_0, \emptyset)) \prod_{i=1}^{t} M_{V'}(v_i, \bar{a}, F(f_0, \emptyset))/Z_n$$

where Z_n is the normalization constant.

Next, we introduce an *iterative Bayes update* model that relaxes the independence assumptions of the Naive Bayes model. The iterative Bayes update model generates a posterior distribution over possible answers at time step t by iteratively applying the vote model on the prior model as given below:

$$Pr(\bar{a}|f) \propto Pr(\bar{a}|F(f_0, h_{t-1}))Pr(< v_t, f_{s_t} > |\bar{a}, F(f_0, h_{t-1}))/Z_b \approx M_A(\bar{a}, F(f_0, \emptyset)) \prod_{i=1}^{t} M_{V'}(v_i, \bar{a}, F(f_0, h_{i-1}))/Z_b$$

where Z_b is the normalization constant.

Another approach is building direct models for predicting the correct answer of a task. A direct model takes as input f, the complete set of features, and predicts $\bar{a}$. Figure 3 shows an example of a direct model trained for predicting the correct answer of a galaxy task.

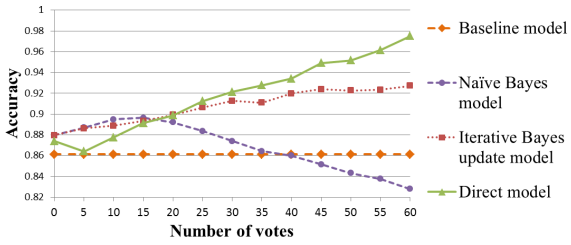


Figure 4: Comparison of accuracies of different models for predicting correct answer.

Figure 4 compares the accuracies of different answer models with a baseline model that classifies every task instance as the most likely correct answer in the training set. Both naive Bayes and iterative Bayes update models perform better than the direct models when the system has a small number of votes. However, the direct models outperform all others as the system collects more votes and as vote features become more predictive of the correct answer. When the system has 45 votes for a galaxy, the accuracy of direct models reach 95%. Based on the significantly stronger performance of the direct models for large numbers of votes, we use direct models in the consensus system.

4.3 Predicting the Next Vote

We also construct a model that predicts the next vote that a system would receive based on task features and worker reports collected so far. This model, symbolized as M_V, takes as input the complete feature set f. It differs from $M_{V'}$ in that the correct answer of a task ($\bar{a}$) is not an input to this model. M_V achieves 65% accuracy when 15 or more votes are collected. We compare the performance of M_V with a baseline model that simply guesses the most likely vote (elliptical galaxy), as the next vote. The comparison shows that M_V has better accuracy than the baseline when 10 or more votes are collected.

4.4 Predicting Termination

Although the system may decide to hire another worker for a task, the execution on a task may stochastically terminate because the system may run out of workers to hire or it may run out of time. Tasks logged in the Galaxy Zoo dataset are associated with different numbers of worker reports. The system has to terminate once all reports for a galaxy are collected. To model the distribution over votes received per task for Galaxy Zoo, we construct a probabilistic termination model from the training set (See Figure 5). The termination model predicts the probability of the system stochastically terminating after collecting different numbers of worker reports.

5. DECISIONS FOR CONSENSUS TASKS

At any time during the execution of the consensus system, the system needs to make a decision about whether to hire an additional worker for each task under consideration. If the system does not hire another worker for a task, it termi-

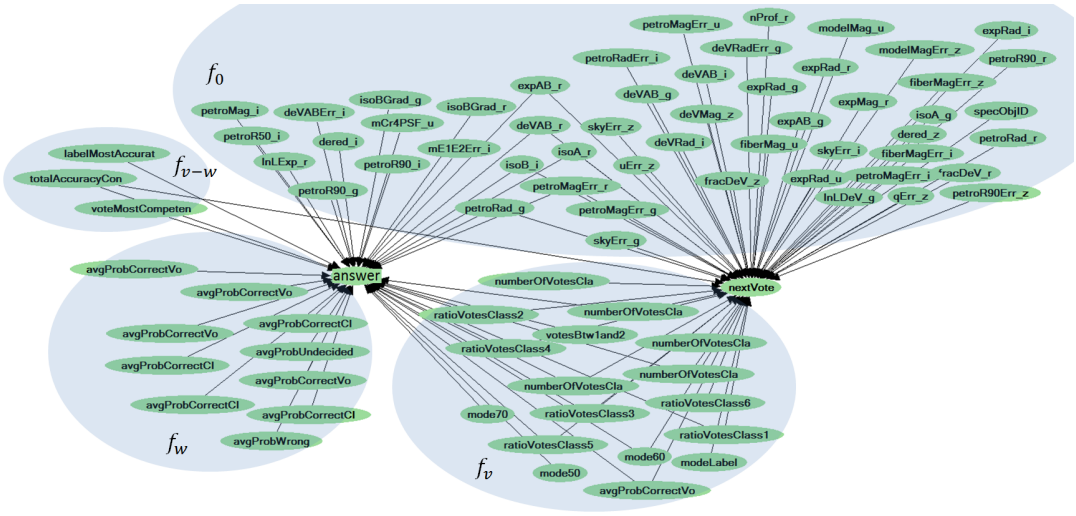


Figure 3: Direct model generated with Bayesian structure learning from Galaxy Zoo data. The model predicts correct answer of a task and next vote that the system would receive.

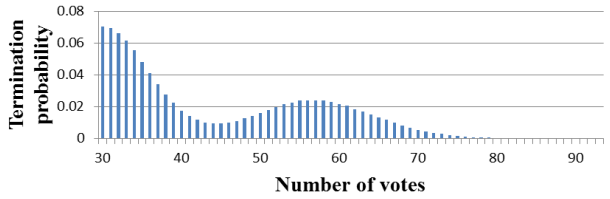


Figure 5: Termination probabilities estimated from training data.

nates and delivers the most likely answer that is predicted by the answer model. If the system decides to hire another worker, it collects additional evidence about the correct answer, which may help the system to predict the answer more accurately. But, hiring a worker incurs monetary and time costs. For solving consensus tasks effectively, the system needs to trade off the long-term expected utility of hiring a worker with the immediate cost. Deliberating about this tradeoff involves the consideration of multiple dimensions of uncertainty. The system is uncertain about the reports it will collect for a given task, and it is not able to observe $\bar{a}$, the correct answer of a consensus task. We shall model this decision-making problem as an MDP with partial observability, which makes calls to the answer and next vote models. We show that exact solutions of consensus tasks over long horizons is intractable and present approximate algorithms for estimating the expected value of hiring a worker.

5.1 Modeling Consensus Tasks

A consensus task is partially observable because the consensus system cannot observe the correct answer of the task. For simplicity of representation, we model a consensus task as an MDP with uncertain rewards, where the reward of the system at any state depends on its belief about the correct answer. A consensus task is formalized as a tuple $< S, \mathbb{A}, T, R, l >$. $s_t \in S$, a state of a consensus task at time t, is composed of a tuple $s_t =< f_0, h_t >$, where f_0 is the set of task features initially available, and h_t is the complete history of worker reports received upto time t.

$\mathbb{A}$, the set of actions for a consensus task include H, hire a worker, and $\neg H$, terminate and deliver the most likely answer to the task owner. $T(s_t, \alpha, s_{t+1})$ is the likelihood of transitioning from state s_t to s_{t+1} after taking action α. The transition function represents the system's uncertainty about the world and about worker reports. The system transitions to a terminal state if the selected action is $\neg H$. If the system decides to hire a worker, the transition probability to a next state depends on likelihoods of worker reports and the likelihood of termination. A worker report is a combination of v_i, worker's vote, and f_{s_i}, the set of features about the worker. To predict the likelihood of a worker report, we use the next vote model, and use average worker statistics computed from the training data to predict f_{s_i}.

The reward function $R(s_t, \alpha)$ represents the reward obtained by executing action α in state s_t. The reward function is determined by the cost of hiring a worker, and the utility function $U(\hat{a}, \bar{a})$, which represents the task owner's utility for the system predicting the correct answer as $\hat{a}$ when it is $\bar{a}$. For the simple case where there is no chance of termination, $R(s_t, H)$ is assigned a negative value which represents the cost of hiring a worker. The value of $R(s_t, \neg H)$ depends on whether the answer that would be revealed by the system based on task features and reports collected so far is correct. b_t is a probability distribution over set A that represents the system's belief about the correct answer of the task, such that for any $a \in A$, $b_t(a) = M_A(a, F(f_0, h_t))$. Let $\hat{a}$ be the most likely answer according to b_t, the reward function is defined as $R(s_t, \neg H) = \sum_{\bar{a}} b_t(\bar{a}) U(\hat{a}, \bar{a})$.

We model consensus tasks as a finite-horizon MDP. l, the horizon of a task, is determined by the ratio of the maximum reward improvement possible (e.g., the difference between the reward for making a correct prediction and the punishment of making an incorrect prediction) and the cost for hiring an additional worker.

A policy π specifies the action the system chooses at any state s_t. An optimal policy π^* satisfies the following equation for a consensus task of horizon l.

$$V^{\pi^*}(s_l) = max_{\alpha \in \mathbb{A}} R(s_l, \alpha)$$
$$V^{\pi^*}(s_t) = max_{\alpha \in \mathbb{A}} \big(R(s_t, \alpha) + \sum_{s_{t+1}} T(s_t, \alpha, s_{t+1})\, V^{\pi^*}(s_{t+1})\big)$$

Now, we can calculate the value of information (VOI) for any given initial state s_i.

$$\begin{aligned} VOI(s_i)) &= V^H(s_i) - V^{\neg H}(s_i) \\ &= R(s_i, H) + \sum_{s_{i+1}} T(s_i, H, s_{i+1})\, V^{\pi^*}(s_{i+1}) \\ &\quad -R(s_i, \neg H) \end{aligned}$$

VOI is the expected value of hiring an additional worker in state s_i. It is beneficial for the consensus system to hire an additional worker when VOI is computed to be positive.

5.2 Solving Consensus Tasks Efficiently

A state of a consensus task at any time step is defined by the history of observations collected for the task. The state space that needs to be searched for computing an optimal policy for a consensus task grows exponentially in the horizon of the task. For large horizons, computing a policy with an exact solution algorithm is infeasible due to exponential complexity. For example, an average Galaxy Zoo task includes 44 worker reports, and the horizon of such a task can be up to 93 time steps.

Myopic decision making and k-step lookahead search are approaches proposed by previous work for approximating the value of information efficiently [5, 4]. These approaches could perform well for solving consensus tasks, if collecting a small set of evidence changed the system's prediction of the correct answer. This condition is unlikely to be satisfied by consensus tasks where worker's reports each provide only weak evidence about the correct answer, and the system needs to reason about the value of collecting a large set of worker reports. For instance, there exists a set of Galaxy Zoo tasks with some particular initial features such that even obtaining 10 consecutive worker reports of the same galaxy label is not enough to change the system's current opinion about the correct answer. Thus, a limited lookahead search has little chance of improving the predictions of the system for this subset of tasks in a reasonable amount of computation time.

Monte-Carlo planning has been used to solve large *fully observable* MDPs [7]. We move to investigate sampling-based solution algorithms, which can be employed in *partially observable* real-world systems for solving consensus tasks accurately and efficiently. These algorithms use Monte-Carlo sampling to perform long lookaheads up to the horizon and to approximate the value of information. Instead of searching a tree that may be intractable in size, this approach samples execution paths (i.e., histories) from a given initial state to a terminal state. For each execution path, it estimates $V^{\neg H}$, the value for terminating at the initial state, and V^H, the value for hiring more workers and terminating later. The value of information is estimated as the difference of these values averaged over a large number of execution path samples. We introduce two algorithms that use this sampling approach to approximate VOI, but differ in the way they estimate V^H. The *lower-bound sampling* (LBS) algorithm picks a single best termination point in the future across all execution paths, V^H is assigned the expected value of this point. The *upper-bound sampling* (UBS) algorithm optimizes the best state for termination for each execution path individually. V^H is estimated by averaging over the values for following these optimal termination strategies. Both algorithms decide to hire an additional worker if VOI is computed to be positive. After hiring a new worker and updating the current state by incorporating new evidence, the algorithms repeat the calculation of VOI for the new initial state to determine whether to hire another worker.

For any given consensus task modeled as an MDP with partial observability, and any initial state s_i, a next state is sampled with respect to the transition function; the likelihood of sampling a state is proportional to the likelihood of transitioning to that state from the initial state. Future states are sampled accordingly until a terminal state is reached. Because sampling of future states is directed by the transition function, the more likely states are likely to be explored. For each state s_t^j on path j, $\hat{a}_t^j$ is the answer predicted based on the current state. When a terminal state is reached, the correct answer for path j, $\bar{a}^j$, is sampled according to the system's belief about the correct answer at this terminal state, when the system is most confident about the correct answer. An execution path from the initial state s_i to a terminal state s_n^j is composed of each state encountered on path j, the corresponding predictions at each state, and the correct answer sampled at the end. It is represented by the tuple: $p^j =< s_i, \hat{a}_i, s_{i+1}^j, \hat{a}_{i+1}^j, ..., s_n^j, \hat{a}_n^j, \bar{a}^j >$.

An execution path represents a single randomly generated execution of a consensus task. For any given execution path, there is no uncertainty about the correct answer or the set of observations that would be collected for the task. Sampling an execution path maps an uncertain task to a deterministic and fully observable execution. To model different ways a consensus task may progress (due to the uncertainty about the correct answer and the worker reports), a library of execution paths (P) is generated by repeating the sampling of execution paths multiple times. This library provides a way to explore long horizons on a search tree that can be intractable to explore exhaustively. If the library includes infinitely many execution paths, it constitutes the complete search tree.

Given an execution path p^j that terminates after collecting n reports, $V_k(p^j)$ is the utility for terminating on this path after collecting k-many worker reports. $V_k(p^j)$ is computed with respect to the answer predicted based on the worker reports collected in the first k steps and the correct answer sampled at the terminal state. Given that c is the cost for hiring a worker, $V_k(p^j)$ is defined as follows:

$$V_k(p^j) = \begin{cases} U(\hat{a}_k^j, \bar{a}^j) - kc & \text{if } k \leq n \\ U(\hat{a}_n^j, \bar{a}^j) - nc & \text{if } n < k \leq l \end{cases}$$

For simplicity of presentation, we assume a constant cost for hiring workers. The definition of $V_k(p^j)$ and consequently LBS and UBS algorithms can be easily generalized to settings in which worker costs depend on the current state.

We define $V^{\neg H}$ with respect to execution path library P as given below:

$$V^{\neg H}(s_i) = \sum_{p^j \in P} V_i(p^j)/|P|$$

The lower-bound sampling (LBS) algorithm approximates V^H as given below:

$$V^H(s_i) = \max_{i<k\leq l}(\sum_{p^j \in P} V_k(p^j)/|P|)$$

LBS picks the value of the best termination step in average for all execution paths. This algorithm underestimates V^H because it picks a fixed strategy for future, and does not optimize future strategies with respect to different worker reports that could be collected in future states. LBS is a pessimistic algorithm; given that the MDP model provided to the algorithm is correct and the algorithm samples infinitely many execution paths, all hire (H) decisions made by the algorithm are optimal.

The upper-bound sampling (UBS) approximates V^H by optimizing the best termination step individually for each execution sequence:

$$V^H(s_i) = \sum_{p^j \in P}(\max_{i<k\leq l} V_k(p^j)/|P|)$$

In distinction to the LBS algorithm, the UBS algorithm overestimates V^H by assuming that both the correct state of the world and future state transitions are fully observable, and thus by optimizing a different termination strategy for each execution sequence. The UBS algorithm is an optimistic algorithm; given that the MDP model provided to the algorithm is correct and the algorithm samples infinitely many execution paths, all not hire ($\neg H$) decisions made by the algorithm are optimal. In the next section, we empirically evaluate the performance of LBS and UBS algorithms on a dataset collected from the Galaxy Zoo system.

6. EXPERIMENTS

We evaluated the ability of the CrowdSynth prototype to guide the solution of consensus tasks on a subset of the testset collected from the Galaxy Zoo project. The testset includes 44350 votes collected for 1000 randomly selected galaxies, and 453 SDSS image features describing each galaxy. We evaluated variations of the system by employing different decision-making algorithms.

The MDP used in CrowdSynth for modeling Galaxy Zoo tasks includes the belief update functions and transition functions learned from real-world data, as described earlier. These predictive models are not perfect; they can be noisy, there can be inconsistencies between consecutive beliefs. This study also helps to evaluate the effect of the noise in the building blocks of an MDP on the performance of different MDP solution algorithms.

6.1 Results

We compare the performance of the decision-theoretic CrowdSynth methodology to two baselines. The first baseline is named *no hire* as it hires no workers and delivers the most likely answer prediction based on the features extracted digitally. The second baseline collects all worker reports available for a task and makes a prediction about the correct answer afterwards. We name this baseline *hire all*. We also implemented myopic and k-step lookahead algorithms, which have been proposed by previous work to estimate VOI. In these experiments, the system is rewarded \$1 for correctly predicting the correct answer of a task (including predicting undecidables), and the cost of hiring a worker is varied between \$0.1 and \$0.0001. The LBS and UBS algorithms used in our investigations terminate after 2000 samples.

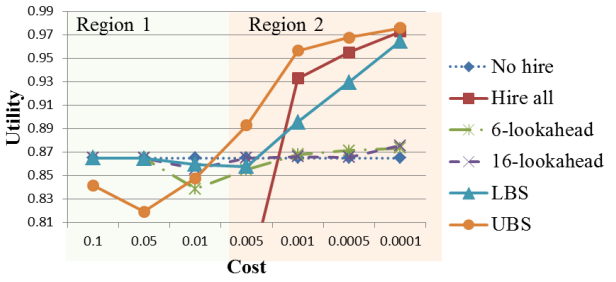


Figure 6: Performance of decision-making models with variation of worker costs.

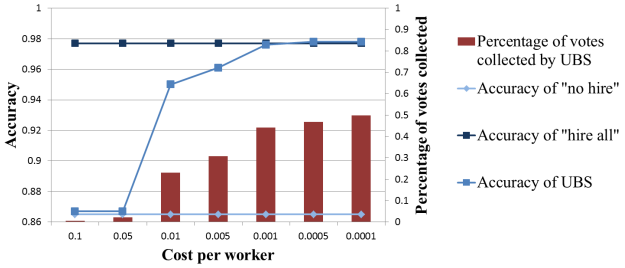


Figure 7: Analysis of behavior of UBS algorithm for varying worker costs.

Figure 6 summarizes the performances of different decision-making algorithms and baselines as a function of the cost of a worker. We divide the figure into two regions of worker costs: high worker costs (Region 1) and low worker costs (Region 2). For high worker costs, none of the decision-theoretic algorithms are able to perform better than the *no hire* baseline, because the expected cost for hiring enough workers to change the system's prediction of the correct answer is as high as or higher the expected benefit.

As shown in Figure 3, predictions of direct answer models are noisier when a few worker reports are collected than when no reports are collected. In Region 1, all decision-theoretic algorithms are affected by this noise because the lookahead depth is relatively short. In addition, the UBS algorithm is affected negatively by the overestimation of VOI in this region.

When the cost of a worker is low, the UBS algorithm performs significantly better than all other algorithms and baselines. The performance of the LBS algorithm is negatively affected by the underestimation of VOI in this region. k-step lookahead algorithms are outperformed by UBS and by LBS (except when cost is 0.005), because for many Galaxy Zoo tasks, even having 16 steps lookahead may not be enough to properly estimate VOI. Overall, the UBS algorithm outperforms the default policy used in the Galaxy Zoo effort (*hire all*) for all worker costs, including high worker costs.

The decision-theoretic approach can perform better than the hire all baseline for varying cost values as it successfully trades off the estimated utility for hiring a worker with the

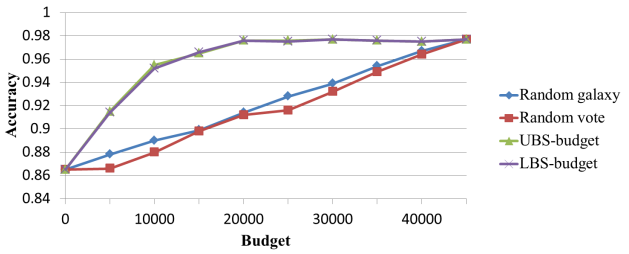


Figure 8: Comparison of decision-making policies under a fixed budget.

cost of doing so. Figure 7 reports the accuracy of the UBS algorithm and the percentage of votes collected by the algorithm for varying cost values. When the cost for hiring a worker is very high, the algorithm hires very few workers (less than 1 worker per celestial object), which results in a slight improvement in accuracy. The algorithm gradually improves its accuracy in predicting the correct answer by hiring more workers as the cost decreases. The algorithm reaches 95% accuracy by collecting only 23% of the reports, it reaches the accuracy of the *hire all* policy by collecting only 47% of the votes. The algorithm is able to improve its accuracy by hiring only a subset of the votes because it can distinguish the set of galaxies for which its decision is likely to change in the future.

For the next set of experiments, we modify the problem so that hiring decisions are not influenced by the cost of hiring a worker, but instead by varying limitations in budget. The budget constrains the total number of workers that can be hired for 1000 celestial objects. The challenge for the system is distributing a limited budget among 1000 different objects to achieve the highest prediction accuracy. We experiment with four decision-making models. The first model, *random galaxy*, randomly selects a celestial object and collects all available votes for that object. The *random vote* model is the approach followed by the original Galaxy Zoo system. This model selects a random object and collects a single vote for that object at each iteration until it runs out of budget. *UBS*-budget and *LBS*-budget models calculate VOI for each celestial object with the UBS and LBS algorithms, and hire a worker with the highest VOI.

Figure 8 compares the performance of these models for varying budgets. Both UBS and LBS models outperform other approaches for all budget sizes. After collecting 20000 votes, the accuracy of the VOI models converge as the system has collected all the evidence it needs to make accurate predictions.

7. DISCUSSION AND FUTURE WORK

We reviewed our efforts to take a principled end-to-end approach to consensus crowdsourcing. We composed a system that combines machine vision, machine learning, and decision-theoretic planning to make effective decisions about when to hire workers and how to perform classifications when observations cease. We constructed a prototype system and evaluated our learning and decision-making techniques on real-world data collected during the operation of the Galaxy Zoo system in the open world. The experiments demonstrate that the methodology can solve consensus tasks accurately and achieve significant savings in worker resources by intelligently allocating resources.

We are exploring extensions of the methods that can reason about optimal timing and pricing of tasks. We have been investigating models that can make predictions about *individual* workers, so that the decision-theoretic planner can make effective decisions about the best worker to hire and the best task to assign to workers who come available. We are also investigating Monte-Carlo approaches that can more accurately estimate VOI. Finally, we are studying challenges with the development of online crowdsourcing services that have the ability to continue to learn from data, combine reports in a coherent manner, and ideally route people and tasks with the Bayesian and decision-theoretic procedures that we have described.

8. ACKNOWLEDGMENTS

We thank Paul Koch for assistance with accessing Galaxy Zoo data, Chris Lintott for sharing the Galaxy Zoo data, and Dan Bohus, Rich Caruana, Paul Koch, and Chris Lintott, for discussions and feedback.

9. REFERENCES

[1] Galaxy Zoo, 2007, http://zoo1.galaxyzoo.org/.

[2] D. Chickering, D. Heckerman, and C. Meek. A Bayesian approach to learning Bayesian networks with local structure. In *UAI'97*, 1997.

[3] P. Dai et al. Artificial intelligence for artificial artificial intelligence. In *AAAI*, 2011.

[4] P. Dai, Mausam, and D. Weld. Decision-theoretic control of crowd-sourced workflows. In *AAAI*, 2010.

[5] D. Heckerman, E. Horvitz, and B. Nathwani. *Toward normative expert systems: The Pathfinder project.* Stanford University, 1991.

[6] P. Ipeirotis. Analyzing the Amazon Mechanical Turk marketplace. *XRDS: Crossroads, The ACM Magazine for Students*, 17(2):16–21, 2010.

[7] L. Kocsis and C. Szepesvári. Bandit based monte-carlo planning. *Machine Learning: ECML 2006*, pages 282–293, 2006.

[8] A. Krause, E. Horvitz, A. Kansal, and F. Zhao. Toward community sensing. In *IPSN*. IEEE, 2008.

[9] C. Lintott et al. Galaxy Zoo: Morphologies derived from visual inspection of galaxies from the Sloan Digital Sky Survey? *Monthly Notices of the Royal Astronomical Society*, 389(3):1179–1189, 2008.

[10] D. Shahaf and E. Horvitz. Generalized task markets for human and machine computation. In *AAAI*, 2010.

[11] V. Sheng, F. Provost, and P. Ipeirotis. Get another label? Improving data quality and data mining using multiple, noisy labelers. In *ACM SIGKDD*, 2008.

[12] R. Snow, B. O'Connor, D. Jurafsky, and A. Ng. Cheap and fast but is it good?: Evaluating non-expert annotations for natural language tasks. In *EMNLP*, 2008.

[13] L. Von Ahn and L. Dabbish. Designing games with a purpose. *Communications of the ACM*, 2008.

[14] J. Whitehill, P. Ruvolo, T. Wu, J. Bergsma, and J. Movellan. Whose vote should count more: Optimal integration of labels from labelers of unknown expertise. *NISP*, 2009.

Reinforcement Learning from Simultaneous Human and MDP Reward

W. Bradley Knox and Peter Stone
Deptartment of Computer Science
The University of Texas at Austin
{bradknox,pstone}@cs.utexas.edu

ABSTRACT

As computational agents are increasingly used beyond research labs, their success will depend on their ability to learn new skills and adapt to their dynamic, complex environments. If human users—without programming skills—can transfer their task knowledge to agents, learning can accelerate dramatically, reducing costly trials. The TAMER framework guides the design of agents whose behavior can be shaped through signals of approval and disapproval, a natural form of human feedback. More recently, TAMER+RL was introduced to enable human feedback to augment a traditional reinforcement learning (RL) agent that learns from a Markov decision process's (MDP) reward signal. We address limitations of prior work on TAMER and TAMER+RL, contributing in two critical directions. First, the four successful techniques for combining human reward with RL from prior TAMER+RL work are tested on a second task, and these techniques' sensitivities to parameter changes are analyzed. Together, these examinations yield more general and prescriptive conclusions to guide others who wish to incorporate human knowledge into an RL algorithm. Second, TAMER+RL has thus far been limited to a *sequential* setting, in which training occurs before learning from MDP reward. In this paper, we introduce a novel algorithm that shares the same spirit as TAMER+RL but learns *simultaneously* from both reward sources, enabling the human feedback to come at any time during the reinforcement learning process. We call this algorithm simultaneous TAMER+RL. To enable simultaneous learning, we introduce a new technique that appropriately determines the magnitude of the human model's influence on the RL algorithm throughout time and state-action space.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Human Factors, Performance

Keywords

reinforcement learning, human-agent interaction, interactive learning, human teachers, shaping

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

1. INTRODUCTION

Computational agents may soon be prevalent in society, and many of their end users will want these agents to learn to perform new tasks. For many of these tasks, the human user will already have significant task knowledge. Consequently, we seek to enable non-technical users to transfer their knowledge to the agent, reducing the cost of learning without hurting the agent's final, asymptotic performance.

In this vein, the TAMER framework guides the design of agents that learn by shaping—using signals of approval and disapproval to teach an agent a desired behavior [7]. As originally formulated, TAMER was limited to learn exclusively from the human feedback. More recently, TAMER+RL was introduced to improve traditional reinforcement learning (RL) algorithms, which learn from an MDP reward signal, with human feedback [8]. However, TAMER+RL has previously only been tested on a single domain, and it has been limited to the case where the learning from human feedback happens only prior to RL: sequential TAMER+RL. We address these limitations by improving upon prior work in two crucial directions.

First, in Section 3, we provide a thorough empirical analysis of the sequential TAMER+RL approach, testing the four TAMER+RL techniques that were previously found to be successful. We test on two tasks—one identical to the single prior TAMER+RL task and a new task. We also provide a much-needed examination of each technique's performance at a range of parameter values to determine the ease of setting each parameter effectively, a critical aspect of using TAMER+RL algorithms in practice that has been previously sidestepped. Together, these analyses yield stronger, more prescriptive conclusions than were possible from prior work. Two similar combination techniques, for the first time, clearly stand out as the most effective, and we consistently observe that manipulating action selection is more effective than altering the RL update.

Second, in Section 4 we introduce a novel algorithm that is inspired by prior work on TAMER+RL but learns from both human and MDP reward simultaneously. The principal benefit of simultaneous learning is its flexibility; it gives a trainer the important ability to step in as desired to alter the course of reinforcement learning while it is in progress. We demonstrate the success of the two best-performing techniques from our sequential experiments, action biasing and control sharing, in this simultaneous setting. To meet demands introduced by the simultaneous setting, we develop a method to moderate the influence of the model of human reward on the RL algorithm. Using this method, simultaneous

TAMER+RL increases the human model's influence in areas of the state-action space that have recently received training and slowly decreases influence in the absence of training, leaving the original MDP reward and base RL agent to learn autonomously in the limit. Without this improvement, the sequential techniques would be too brittle for simultaneous learning.

2. PRELIMINARIES

In this section, we briefly introduce reinforcement learning and the TAMER Framework.

2.1 Reinforcement Learning

We assume that the task environment is a Markov decision process (MDP) specified by the tuple (S, A, T, γ, D, R). S and A are respectively the sets of possible states and actions. T is a transition function, $T : S \times A \times S \rightarrow \mathbb{R}$, which gives the probability, given a state s_t and an action a_t, of transitioning to state s_{t+1}. γ, the discount factor, exponentially decreases the value of a future reward. D is the distribution of start states. R is a reward function, $R : S \times A \times S \rightarrow \mathbb{R}$, where the reward is a function of s_t, a_t, and s_{t+1}. We will also consider reward that is a function of only s_t and a_t.

Reinforcement learning algorithms (see Sutton and Barto [15]), seek to learn policies ($\pi : S \rightarrow A$) for an MDP that maximize return from each state-action pair, where return is $\sum_{t=0}^{T} E[\gamma^t R(s_t, a_t, s_{t+1})]$. In this paper, we focus on using a value-function-based RL method, namely SARSA(λ) [15], augmented by the TAMER-based learning that can be done directly from a human's reward signal. Though more sophisticated RL methods exist, we use SARSA(λ) for its popularity and representativeness, and because we are not concerned with finding the best overall algorithm for our experimental tasks but rather with determining how various techniques for including a human model change the base RL algorithm's performance.

2.2 The TAMER Framework for Interactive Shaping

The TAMER Framework [7] is an approach to the problem of how an agent should learn from numerically mapped reward signals given by a human trainer. Specifically, these feedback signals are delivered by an observing human trainer as the agent attempts to perform a task.[1] TAMER is motivated by two insights about human reward. First, human reward is trivially delayed, slowed only by the time it takes the trainer to assess behavior and deliver feedback. Second, the trainer observes the agent's behavior with a model of that behavior's long-term effects, so the human reward signal is assumed to be fully informative about the quality of recent behavior. Human reward is more similar to an action value (sometimes called a Q-value), albeit a noisy and trivially delayed one, than MDP reward. Consequently, TAMER assumes human reward to be fully informative about the quality of an action given the current state, and it models a hypothetical human reward function, $H : S \times A \rightarrow \mathbb{R}$, as $\hat{H}$ in real time by regression. In the simplest form of credit assignment, each human reward signal creates a label for the last state-action pair.[2] The output of the resultant $\hat{H}$ function—changing as the agent gains experience—determines the relative quality of potential actions, so that the exploitative action is $a = argmax_a[\hat{H}(s, a)]$.

[1] In our experiments, the trainer has a button for positive reward and one for negative. Multiple button presses are roughly interpreted as more intense feedback.

3. SEQUENTIAL TAMER+RL

Observing that TAMER agents typically learn faster than agents learning from MDP reward but to a lower performance plateau, we combined TAMER and SARSA(λ) in the original publication on TAMER+RL [8]. The aim was to complement TAMER's fast learning with RL's ability to often learn better policies in the long run. These conjoined TAMER+RL algorithms address a scenario in which a human trains an agent, leaving a model $\hat{H}$ of human reward, and then $\hat{H}$ is used to influence the base RL algorithm somehow. We call this scenario and the algorithms that address it *sequential* TAMER+RL. For all TAMER+RL approaches, only MDP reward is considered to specify optimal behavior. $\hat{H}$ provides guidance but not an objective. In this section, we reproduce and then extend prior investigations of sequential TAMER+RL, yielding more prescriptive and general conclusions than prior work allowed.

3.1 Combination techniques

Eight TAMER+RL techniques were previously tested [8]; each uses $\hat{H}$ to affect the RL algorithm in a different way. Four were largely effective when compared to the SARSA(λ)-only and TAMER-only agents[3] on both mean reward over a run and performance at the end of the run. We focus on those four techniques, which can be used on any RL algorithm that uses an action-value function. Below, we list them with names we have created.[4] In our notation, a prime (e.g., Q') after a function means the function replaces its non-prime counterpart in the base RL algorithm.

- **Reward shaping**: $R'(s, a) = R(s, a) + (\beta * \hat{H}(s, a))$
- **Q augmentation**: $Q'(s, a) = Q(s, a) + (\beta * \hat{H}(s, a))$
- **Action biasing**: $Q'(s, a) = Q(s, a) + (\beta * \hat{H}(s, a))$ *only during action selection*
- **Control sharing**: $P(a{=}argmax_a[\hat{H}(s, a)]) = min(\beta, 1)$. *Otherwise use base RL agent's action selection mechanism.*

In the descriptions above, β is a predefined *combination parameter*. In our sequential TAMER+RL experiments, β is annealed by a predefined factor after each episode for all techniques other than Q augmentation.

We now briefly discuss these techniques and situate them within related work. In the RL literature, reward shaping adds the output of a shaping function to the original MDP reward, creating a new reward to learn from instead [3, 10]. As we confirm in the coming paragraph on Q augmentation, the reward shaping technique used in this paper is not the only way to do reward shaping, though it is the most direct use of $\hat{H}$ for reward shaping.

[2] The trivial delay is dealt with using a credit assignment technique described previously [7].

[3] A TAMER-only agent simply uses $\hat{H}$ to choose actions, ignoring MDP reward. In sequential TAMER+RL, $\hat{H}$ is constant, and thus so is the agent's policy.

[4] These four techniques are numbered 1, 4, 6, and 7 in past work [8]. We altered action biasing to generalize it, but the ϵ-greedy policies we use in our experiments are not affected.

If $\hat{H}$ is considered a heuristic function, action biasing is the same action selection method used in Bianchi et al.'s Heuristically Accelerated Q-Learning (HAQL) algorithm [1]. Control sharing is equivalent to Fernández and Veloso's π-*reuse exploration strategy* [4]. Note that both control sharing and action biasing only affect action selection and can be interpreted as directly guiding exploration toward human-favored state-action pairs.

Q augmentation is action biasing with additional use of $\hat{H}$ during the Q-function's update. Wiewiora et al.'s related *look-ahead advice* [20] uses a discounted change in the output of a state-action potential function, $\gamma\phi(s_{t+1}, a_{t+1}) - \phi(s_t, a_t)$, for reward shaping and to augment action values during action selection. Interestingly, *look-ahead advice* is equivalent to Q augmentation when $\hat{H}$ is used for ϕ, the state and action space are finite, and the policy is invariant to adding a constant to all action values in the current state (e.g., ϵ-greedy and soft-max).

3.2 Sequential learning experiments

We now describe our sequential TAMER+RL experiments. We first reproduce results on the single task on which the techniques were previously tested.[5] We then evaluate the algorithms' effectiveness on a different task. Additionally, we analyze our results at a range of combination parameter values (β values) to identify challenges to setting β's value without prior testing.

Using the original $\hat{H}$ representation (linear model of RBF features), task settings, SARSA(λ) parameters, and training records,[6] we repeat past experiments on the mountain-car task, using all four combination techniques found to be successful in those experiments and a range of β combination parameters. We then test these TAMER+RL techniques on a second task, cart pole, using an $\hat{H}$ model trained by an author. We again use SARSA(λ), choosing parameters that perform well but sacrifice some performance for episode-to-episode stability and the ability to evaluate policies that might otherwise balance the pole for too long to finish a run. Both tasks are adapted from RL-Library [16]. In mountain car, the goal is to quickly move the car up a hill to the goal. The agent receives -1 reward for all transitions to non-absorbing states. In cart pole, the goal is to move a cart so that an attached, upright pole maintains balance as long as possible. The agent receives +1 reward for all transitions that keep the pole within a specified range of vertical. The $\hat{H}$ for cart pole was learned by k-Nearest Neighbor. For both tasks, SARSA(λ) uses a linear model with Gaussian RBF features and initializes Q pessimistically, as was found effective previously [8]. In these and later experiments, $\hat{H}$ outputs are typically in the range [-2, 2].

We evaluate each combination technique on *four criteria*; full success requires outperforming the corresponding $\hat{H}$'s TAMER-only policy and SARSA(λ)-only both in end-run performance and cumulative reward (or mean reward across full runs, equivalently).

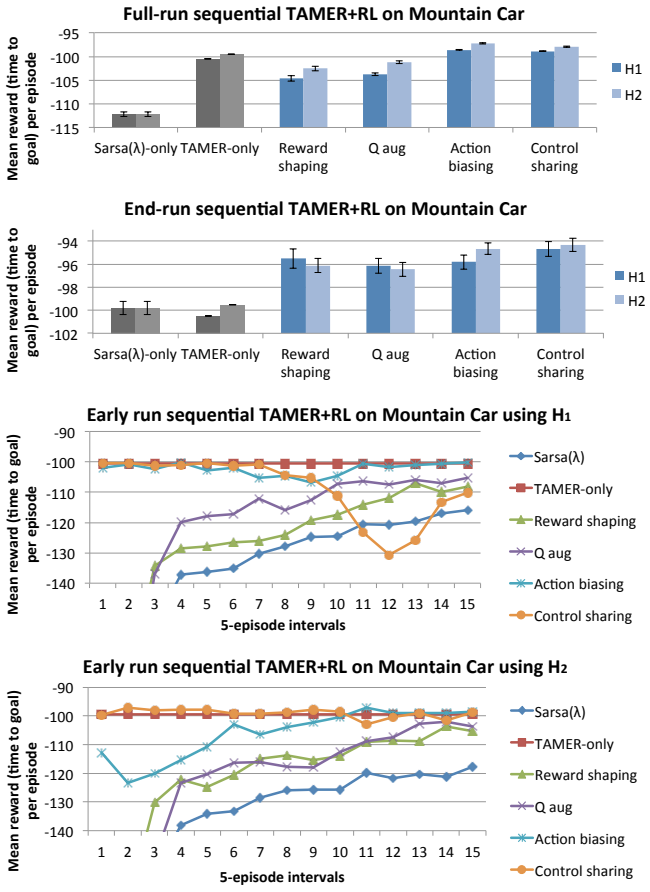


Figure 1: Comparison of TAMER+RL techniques with SARSA(λ) and the TAMER-only policy on mountain car over 40 or more runs of 500 episodes. $\hat{H}_1$ and $\hat{H}_2$ are models from two different human trainers. The top chart considers reward over the entire run, and the second chart evaluates reward over the final 10 episodes. Error bars show standard error. The third and fourth charts display mean performance using $\hat{H}_1$ and $\hat{H}_2$ early in the run, during the first 75 episodes.

3.3 Sequential learning results and discussion

Figures 1 and 2 show the results of our experiments for sequential TAMER+RL. For now, we only show results for the β combination parameters that accrue the highest cumulative reward for their corresponding technique. Figure 2 additionally shows learning curves for the first 30 episodes of the cart pole run.

Qualitatively, our mountain car results are consistent with previous work. Action biasing and control sharing succeed on all four criteria and significantly outperform other techniques in cumulative reward. Reward shaping and Q augmentation also improve over SARSA(λ)-only by both metrics and over the TAMER-only policies in end-run reward.

On cart pole, action biasing and control sharing again suc-

[5]The experiments described in this paper use a different implementation of TAMER than was used in previous work [7]. This version has minor algorithmic differences and one significant change in the credit assignment technique, which we we will not fully describe here for space considerations. Briefly, for each human reward signal received, past TAMER algorithms created a learning sample for every time step within a window of recent experience, resulting in many samples per human reward signal in fast domains. We instead create one sample per time step, using all crediting rewards to create one label.

[6]The models we create—$\hat{H}_1$ and $\hat{H}_2$—from the original training trajectories perform a bit better than those from previous experiments [8], which points to small implementation differences.

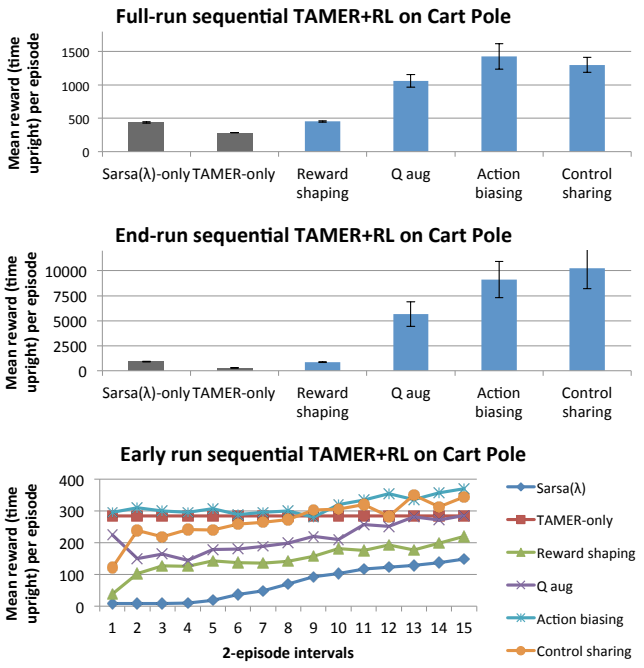


Figure 2: The same TAMER comparisons as in Figure 1, except on cart pole over runs of 150 episodes. A single $\hat{H}$ was used. End-run performance for cart pole is the mean reward during the last 5 episodes.

ceed fully. This time, Q augmentation also meets the four criteria for success, though it performs significantly worse than action biasing and control sharing. Most interestingly, reward shaping, at its best tested parameter, does not significantly alter SARSA(λ)'s performance on either metric.

By choosing the best β parameter value for each technique, prior TAMER+RL experiments sidestep the issue of using an effective value without first testing a range of values. With experiments in two tasks, we can begin to address this problem by examining each technique's sensitivity to β parameter changes and whether certain ranges of β are effective across different tasks. In Figure 3, we show the mean performance of each combination technique as β varies. Examining the charts, we consider several criteria:

- performance at worst β value,
- range of beneficial β values,
- and existence of β values that are effective across tasks.

Evaluating the techniques on these three criteria creates a consistent story that fits with our analysis of the techniques at their best β parameter values (in Figures 1 and 2). The two combination methods that only affect action selection—action biasing and control sharing—emerge as the most effective techniques without a clear leader between them, and they are followed by Q augmentation and then shaping rewards.

From an RL perspective, the weakness of reward shaping may be counterintuitive. When researchers discuss combining human reward with RL in the literature, reward shaping is predominantly suggested [19, 5], possibly because human "reward" is seen as an analog to MDP reward that should be used similarly. However, though reward shaping is generally cast as a guide for exploration, it only affects exploration indirectly through precariously tampering with the reward signal. Action biasing and control sharing affect exploration directly, without manipulating reward. Thus, they achieve the stated goal of reward shaping while leaving the agent to learn accurate values from its experience. Following this line of thought, Q augmentation is identical to action biasing during action selection, boosting each action's Q-value by the weighted prediction of human reward. In addition to this direct guidance on exploration, Q augmentation also changes the Q-value during the SARSA(λ) update's calculation of temporal difference error. As discussed in Section 3.1, Q augmentation is nearly equivalent to a form of reward shaping called look-ahead advice [20]. In short, we observe that the more a technique directly affects action selection, the better it does, and the more it affects the update to the Q function for each transition experience, the worse it does. Q augmentation does both and performs between the techniques that do only one.

Taken altogether, these experiments validate the conclusions of past TAMER+RL work and yield new, firmer conclusions about the relative effectiveness of each technique, endorsing action biasing and control sharing over the two other previously successful techniques. And more generally, these results endorse manipulating action selection and leaving the action-value model's update unmolested.

4. SIMULTANEOUS TAMER+RL

To this point, similarly to all prior work on TAMER, we have assumed that the human training was finished prior to any reinforcement learning. This "sequential" learning is sometimes appropriate; for instance, when a difficult-to-simulate reward function is tied to potentially costly learning trials and the agent can train in simulation without significant cost. However, in other scenarios this assumption can be limiting. In this section, we investigate how to modify sequential TAMER+RL algorithms to allow a trainer to step in as desired to alter the course of reinforcement learning while it is in progress. We call this scenario and the algorithms that address it "simultaneous" TAMER+RL. Specifically, the agent should learn simultaneously from two feedback modalities—human reward and MDP reward—as one fully integrated system. As in the sequential TAMER+RL approaches, we examine techniques that use only $\hat{H}$ from TAMER in the RL algorithm, otherwise leaving the two algorithms as separate modules.

Since TAMER empirically compares most favorably against RL algorithms in early learning [7], we expect the greatest gains to come from training near the beginning of learning. However, training at any suboptimal point along the learning curve should benefit the agent, and we hope to do little harm if the agent is already performing optimally and the trainer's feedback cannot help.

Some desirable characteristics for simultaneous learning are:

1. *steady behavior*: When the agent's behavior changes frequently, giving quality feedback becomes more difficult.
2. *responsiveness to the trainer*: The agent should quickly and obviously demonstrate that it is learning from human reward to maintain interactivity. Additionally, quick responses aid a trainer's own process of learning how to teach effectively.

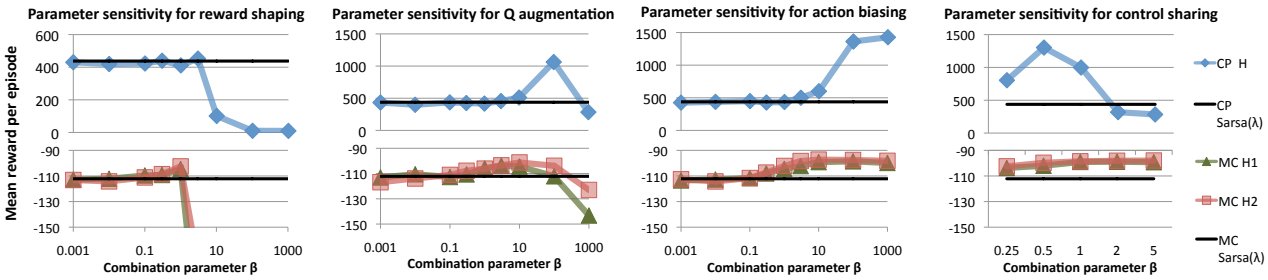


Figure 3: Performance of each technique with each tested $\hat{H}$ over ranges of β parameters on two tasks: cart pole (CP) and mountain car (MC). Note changes in y-axis scaling and that while $\beta \geq 1$ control sharing always chooses by $\hat{H}$.

3. *trainer can give feedback to the MDP-only policy*: If a trainer comes in midway through learning, the trainer should be able to *capture* the good aspects of what has already been learned and criticize the negative aspects.
4. *trainer's influence is applied appropriately*: $\hat{H}$'s influence on the RL algorithm's learning and/or action selection should be larger in more recently trained areas of the state-action space and smaller in areas trained less recently.

Simultaneous learning—and its inclusion of RL-based action selection during training—presents new challenges for maintaining behavioral consistency. For instance, control sharing abruptly shifts between two policies, which can create erratic behavior with many different actions (both good and bad) in a small time period, increasing the difficulty of giving clear feedback. Also note that the second and third characteristics are in opposition. Fully responding to the trainer's reward requires abandoning the policy learned by MDP reward. Our module for determining human influence, described in the following section, strikes a balance by ramping up the influence of $\hat{H}$ with increased human reward, keeping the RL policy early on.

4.1 Determining the immediate influence of $\hat{H}$

Simultaneous TAMER+RL allows humans trainers to insert themselves at any point of the learning process. Consequently, $\hat{H}$'s influence should increase in areas of the state-action space with recent human training—but not in areas that have not been targeted with feedback—and decrease in the absence of training, leaving the set of optimal policies unchanged in the limit.[7] Thus, we must do more than annealing a combination parameter, as is done in sequential learning.

We determine $\hat{H}$'s influence through a novel adaptation of the eligibility traces often used in reinforcement learning [15]. We will refer to it as the *eligibility module*. Watching the demonstration of simultaneous TAMER+RL at `http://cs.utexas.edu/~bradknox/simultamerrl` may be helpful prior to reading the details below. The general idea of this eligibility module is that we maintain an eligibility trace for each state-action feature[8], normalized between 0 and 1, that represents the recency of training while that feature was active (i.e., non-zero). Then, the eligibility traces and a time step's feature vector together calculate a measure of the recency of training in similar feature vectors, as shown in Figure 4. That measure, multiplied by a constant scaling parameter c_s, is used as the β term introduced in Section 3.1. The implementation follows.

Let $\mathbf{e}$ be the vector of traces and $\mathbf{f_n}$ be the feature vector normalized such that each element of $\mathbf{f_n}$ exists within the range $[0, 1]$. The eligibility module is designed to make β a function of $\mathbf{e}$, $\mathbf{f_n}$, and c_s with range $[0, c_s]$. A guiding design constraint is that when $\mathbf{e} = \mathbf{1}$ (i.e., each element of $\mathbf{e}$ is the maximum allowed), the normalized dot product of $\mathbf{e}$ and any $\mathbf{f_n}$, denoted $n(\mathbf{e}{\cdot}\mathbf{f_n})$, should equal 1 (since it weights the influence of $\hat{H}$). To achieve this, we make $n(\mathbf{e}{\cdot}\mathbf{f_n}) = \mathbf{e} \cdot (\mathbf{f_n} \,/\, \| \mathbf{f_n} \|_1) = (\mathbf{e}{\cdot}\mathbf{f_n}) \,/\, (\| \mathbf{f_n} \|_1) = \beta \,/\, c_s$. Thus, at any time step with normalized features $\mathbf{f_n}$, the influence of $\hat{H}$ is calculated as $\beta = c_s(\mathbf{e}{\cdot}\mathbf{f_n})/(\| \mathbf{f_n} \|_1)$. This formula has a desirable mathematical characteristic; for a given $\mathbf{e}$, β is higher when relatively large feature values correspond to large trace values—indicating the current state-action pair is similar to the recently trained state-action pairs—and β is smaller when large feature values correspond to small trace values.

Using accumulating traces capped at 1, the trace is updated with $\mathbf{f_n}$ during training: $e_i := min(1, e_i + (f_{n,i} * a))$, where e_i and $f_{n,i}$ are the i^{th} elements of $\mathbf{e}$ and $\mathbf{f_n}$, respectively, and a is a constant factor that moderates the speed of accumulation. During time steps without training, $\mathbf{e} := decayFactor * \mathbf{e}$.

Though this eligibility module is inspired by eligibility traces used in TD(λ), it differs from eligibility traces in several key ways. This module maintains a vector of traces similarly to how TD(λ)'s eligibility traces are maintained. However, unlike eligibility traces, it only increases the traces during training. In addition, rather using the traces to determine the extent that each feature's corresponding Q-value parameter is updated, we use them to output a measure that roughly indicates how recently nearby states have been trained.

4.2 Simultaneous learning experiments

Our experiments test the effectiveness of simultaneous TAMER+RL when training starts either at the beginning of learning or after some learning has occurred. We again use mountain car and cart pole, and we focus on the two best-performing combination techniques, action biasing and control sharing.

The eligibility module's features are Gaussian RBFs that

[7] Our approach is designed to have the qualitative characteristics we see as necessary for simultaneous learning; we doubt any notions of theoretical "correctness" can be assessed without brittle assumptions about the human

[8] The feature vector is extracted from the current state-action pair. We advise using features that generalize across state space (e.g., Gaussian RBFs). The state-action features need not match those of either $\hat{H}$ or Q.

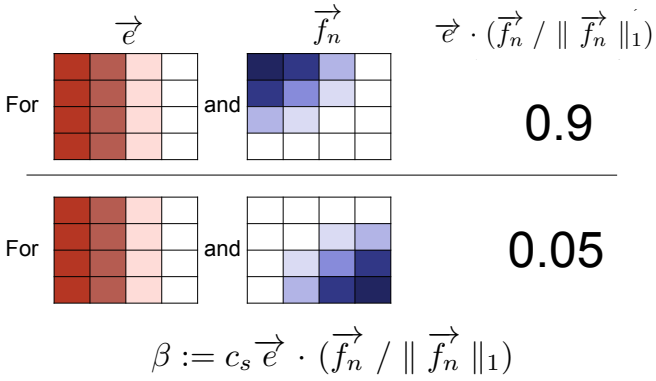


Figure 4: A simple graphic illustration of the calculation of $\mathbf{e} \cdot (\mathbf{f_n} \,/\, \| \mathbf{f_n} \|_1) = (\mathbf{e} \cdot \mathbf{f_n}) \,/\, (\| \mathbf{f_n} \|_1)$, which is near 0 when the currently "active" features have not been active during recent training and is near 1 when these features have been. Here, consider $\mathbf{f_n}$ to be a 4×4 set of Gaussian radial basis functions that form a grid over a 2-dimensional state space. (For simplicity, the action is not considered here.) In the top scenario, the state is somewhere in the top left square and the active features overlap heavily with recently trained state. Thus, the output is near one, possibly 0.9. In the bottom scenario, the state is in the bottom-right square where there is less overlap, resulting in a lower output such as 0.05.

are extracted similarly to the SARSA(λ) features. Also, β's application in control sharing cannot be action-specific, so the eligibility module's features for control sharing are a single grid over the state space (not one grid per action as for SARSA(λ) and for action biasing's eligibility module). In the eligibility module, the scaling parameter c_s for mountain car and cart pole is respectively 100 and 200 for action biasing and 2 and 1 for control sharing. These values were chosen to be near the upper end of each combination method's effective β values in Figure 3. The accumulation factor a for eligibility is 0.2. Training in mountain car occurs either for 16 episodes, starting at episode 1, or for 12 episodes after 20 episodes of SARSA(λ)-only learning. In cart pole, training at start occurs for 12 episodes, and training after 25 episodes of SARSA(λ)-only learning lasts 8 episodes. The start times are chosen to represent the beginning of learning and also a point at which the SARSA(λ) agent has learned a policy that is much improved but still quite flawed.[9] The number of episodes corresponds to an informal assessment of how many episodes are needed to satisfactorily train the agent; training at later start times progresses more quickly. The trainer, one of the authors, has a button that starts and stops training during the designated training episodes, letting the human observe without the agent updating $\hat{H}$ or increasing any traces within the eligibility module. At all times, whether training is occurring or not, the agent continues to learn a Q-function.

An added experimental challenge is that the training is inextricably bound to one specific run, whereas sequential

[9]Note that sequential TAMER+RL differs from simultaneous TAMER+RL where training occurs at the start because the sequential algorithm begins with a pre-trained $\hat{H}$. The training episodes are not counted in sequential TAMER+RL experiments.

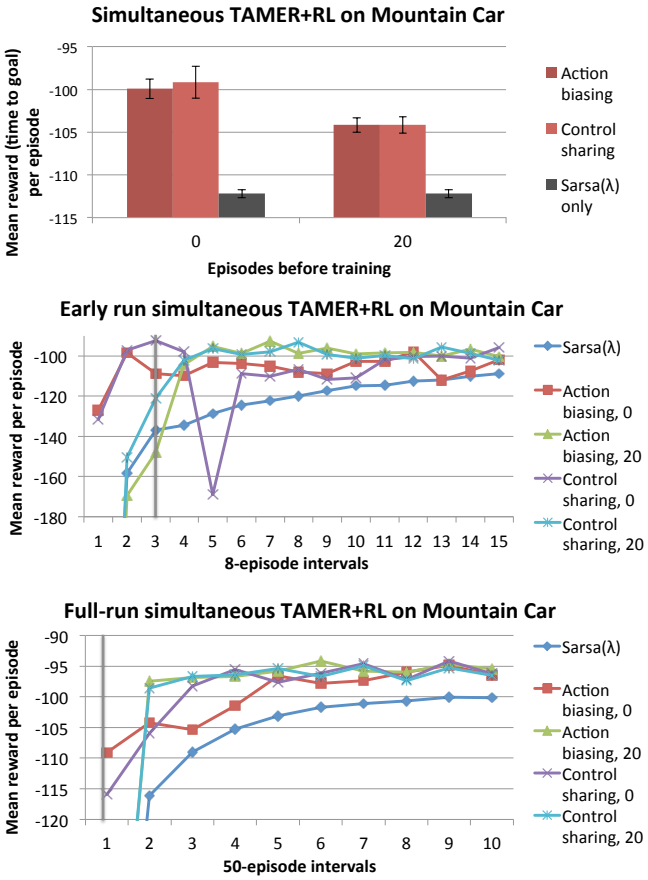


Figure 5: Simultaneous TAMER+RL results on mountain car. Unlike sequential TAMER+RL, performance during training episodes is counted. In the top graph, mean reward is calculated over runs of 500 episodes in mountain car and 150 episodes in cart pole. Standard error is shown. In the lower two plots, learning curves are shown at two different scales: the top plot shows mean performance in the earlier episodes of the run and the bottom plot shows mean performance over the entire run. The number in each legend entry indicates the episode number at which training started. A vertical gray bar is placed at the point where the later training period started, the 20^{th} episode.

experiments can reuse the same training session for any number of parameters and combination techniques, limiting the depth of analysis that can be done for a set number of trainer-hours. Mountain car and cart pole training sessions typically took around 8 minutes and 15 minutes each, respectively. Consequently, each experimental condition was limited to 3 runs of training for a total of 12 runs on each task.

4.3 Simultaneous learning results and discussion

The results of our simultaneous TAMER+RL experiments are shown in Figures 5 and 6. Though the sample size is too small to show statistical significance, there is a clear pattern of both action biasing and control sharing outperforming SARSA(λ). The condition that is closest to SARSA(λ) in terms of standard error, control sharing on cart pole where

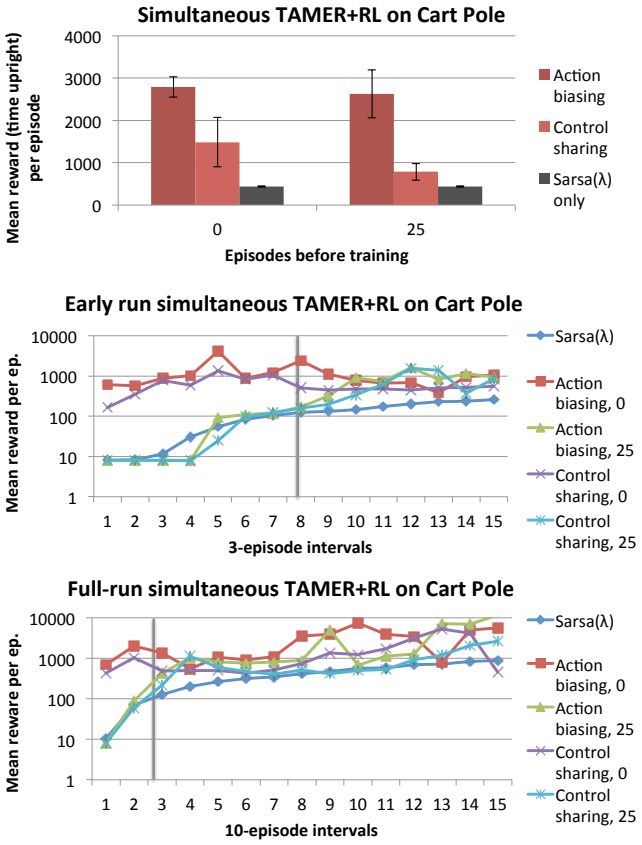


Figure 6: Learning curves for simultaneous TAMER+RL on cart pole, following the same format as Figure 5.

training begins after 25 episodes, still receives almost twice the reward of SARSA(λ). We also observe that training at the beginning of learning is more effective—in terms of mean reward during a run—than training after some MDP-only learning, as we expected.

Seeing that training is most effective at the start of learning, one might ask whether the n episodes of MDP-only learning before training is helping or whether the prior learning should be abandoned to start from scratch. We can quantitatively evaluate this question. Starting from scratch after n episodes is the same as simply training from the start and stopping n episodes early. So if we ignore the first n episodes of the later-training group and the last n episodes of the training-at-start group, the comparison of the groups' mean reward addresses this question. In other words, for a task with run size m, we examine two conditions per combination technique: (1) from the trajectories where training began at the first episode, the performance of the the agent from episodes $(1, 2, ..., m-n)$ is averaged, and (2) from the trajectories where training began at episode n, the performance of the agent during episodes $(n+1, n+2, ..., m)$ is averaged. Thus, each condition is examined over $m-n$ episodes, and training begins at the first episode of examination. The main difference between the conditions is that the later-training group (i.e., starting at n) starts training after already learning from MDP reward, so we can reasonably conclude that performance differences arise from the presence or lack of MDP-only learning prior to training.

Of four such comparisons (2 techniques x 2 tasks, shown in Figure 7), the later-training group outperforms three times and is roughly equal once, suggesting that the prior learning does indeed help.[10]

For clarity, we note that we do not aim to quantitatively compare sequential and simultaneous TAMER+RL. Our results in Figure 7 conclusively show the benefit of training after some MDP-only learning, when it is too late to learn sequentially. Therefore, simultaneous learning provides benefits that sequential learning cannot. And when training without MDP reward is relatively costless, sequential learning allows an agent to be thoroughly taught before beginning more costly learning with MDP reward; thus, sequential learning likewise provides benefits that simultaneous learning cannot. Neither learning scenario is strictly better than the other.

These results, shown in Figures 5, 6, and 7, demonstrate the potential effectiveness of simultaneous TAMER+RL with the eligibility module.

5. RELATED WORK

In this section, we situate our work within prior research on naturally transferring knowledge to a reinforcement learning agent. We focus on work not already mentioned in Section 3.1 or in the previous papers on TAMER [7, 8].

Sridharan [13] also extended the original TAMER+RL work [8], showing that action biasing improves an RL agent's learning in the domain of 3v2 keepaway. In this work, he tests a "bootstrapping" approach for determining β that sets it by a metric of agreement between the Q-function policy and the $\hat{H}$-policy, where more agreement yields a larger β.

In the only other algorithm for an agent learning simultaneously from both human and MDP reward [19], Thomaz and Breazeal interfaced a human trainer with a table-based Q-learning agent in a virtual kitchen environment. Their agent seeks to maximize its discounted total reward, which for any time step is the sum of human reward and MDP reward. Their approach is a form of reward shaping, differing in that Thomaz and Breazeal directly apply the human reward value to the current reward (instead of modeling human reward and using the output of the model as supplemental reward). Tenorio-Gonzalez et al. [18] expanded this learning algorithm, additionally using human demonstrations. In their algorithm—tested on a trainer and a virtual robot—the RL agent learns from (s, a, r, s') experiences created during demonstrations.

Judah et al. consider a learning scenario that alternates between "practice", where actual world experience is gathered, and an offline labeling of actions as good or bad by a human critic [6]. Using an elegant probabilistic technique with a few assumptions, the human criticism is input to a loss function that lessens the expected value of candidate policies while also automatically determining the level of influence given to the criticism. From some mixed results and comments from frustrated subjects, they predicted that redesigning their system to be more interactive and to let

[10]The only other known difference between conditions is that agents with prior learning were given less episodes of training than those that were trained, as was described earlier in this section. Despite the apparent disadvantage of fewer training episodes, these agents still outperform those without prior learning in the subset of episodes examined here.

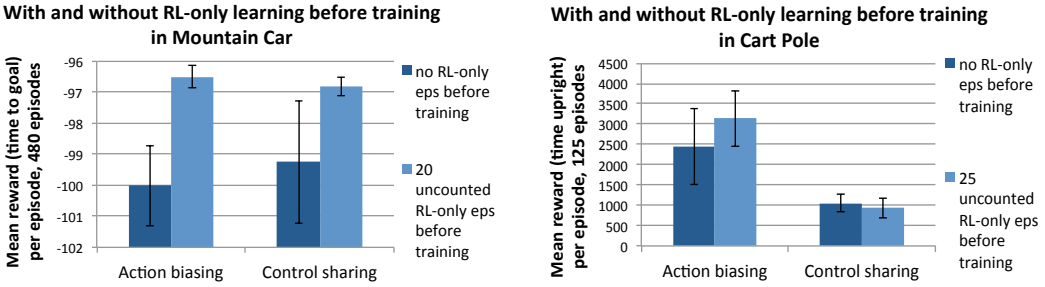


Figure 7: A comparison of simultaneous TAMER+RL performance with and without MDP-only learning before training. The lighter blue bars indicate the mean performance of agents that received 20 or 25 episodes of MDP-only learning before the human began teaching. The episodes of prior learning are not counted towards mean reward. The agents corresponding to darker blue bars had no such learning prior to training. Standard error is shown.

the human train periodically—characteristics of simultaneous TAMER+RL—would improve performance.

Learning from demonstration (LfD) has also been used to improve reinforcement learning, using preprogrammed policies [11] or humans [17, 12] to provide demonstrations for an agent that observes and learns. These approaches are similar to control sharing. An advantage, though, of human reward over demonstration is that human reward permits learning the relative values of actions, allowing techniques like action biasing to gently push the behavior of the RL agent towards the policy endorsed by $\hat{H}$, whereas pure demonstration is all or nothing—either the demonstrator or the learning agent chooses the action. Additionally, trainers can reward state-action pairs visited by the agent's policy, whereas demonstrations might not ever visit areas of the state space that the LfD algorithm visits.

Other recent work has employed non-expert humans to aid RL algorithms through guidance on feature selection [2], identification and demonstration of high-level actions [14], and giving natural language advice [9].

6. CONCLUSION

Prior work on TAMER+RL is limited by having only been tested on a single domain and by simply taking the best β combination parameter from testing. Further, past TAMER+RL algorithms were designed for sequential learning and were unsuitable for simultaneously learning from the trainer and the MDP reward signal. This paper addresses these limitations, giving a clear endorsement of using $\hat{H}$ to affect action selection and, for the first time, enabling a human trainer to interactively provide feedback at any time during the learning process, a critical improvement towards the practicality and widespread applicability of the TAMER framework.

Acknowledgments

This work has taken place in the Learning Agents Research Group (LARG) at The University of Texas at Austin. LARG research is supported in part by grants from the NSF (IIS-0917122), ONR (N00014-09-1-0658), and the FHA (DTFH61-07-H-00030). W. Bradley Knox has been supported by an NSF Graduate Research Fellowship.

7. REFERENCES

[1] R. Bianchi, C. Ribeiro, and A. Costa. Heuristically Accelerated Q–Learning: a new approach to speed up Reinforcement Learning. *Advances in AI – SBIA*, 2004.

[2] L. Cobo, P. Zang, C. Isbell Jr, and A. Thomaz. Automatic state abstraction from demonstration. In *IJCAI*, 2011.

[3] M. Dorigo and M. Colombetti. Robot shaping: Developing situated agents through learning. *Artificial Intelligence*, 1994.

[4] F. Fernández and M. Veloso. Probabilistic policy reuse in a reinforcement learning agent. *AAMAS*, 2006.

[5] C. Isbell, M. Kearns, S. Singh, C. Shelton, P. Stone, and D. Kormann. Cobot in LambdaMOO: An Adaptive Social Statistics Agent. *AAMAS*, 2006.

[6] K. Judah, S. Roy, A. Fern, and T. Dietterich. Reinforcement Learning Via Practice and Critique Advice. *AAAI*, 2010.

[7] W. Knox and P. Stone. Interactively shaping agents via human reinforcement: The TAMER framework. *K-CAP*, 2009.

[8] W. Knox and P. Stone. Combining manual feedback with subsequent MDP reward signals for reinforcement learning. *AAMAS*, 2010.

[9] G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik. Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *The AAAI-2004 Workshop on Supervisory Control of Learning and Adaptive Systems*, July 2004.

[10] M. Mataric. Reward functions for accelerated learning. *ICML*, 1994.

[11] B. Price and C. Boutilier. Accelerating reinforcement learning through implicit imitation. *JAIR*, 19:569–629, 2003.

[12] W. Smart and L. Kaelbling. Practical reinforcement learning in continuous spaces. *ICML*, 2000.

[13] M. Sridharan. Augmented reinforcement learning for interaction with non-expert humans in agent domains. In *Proceedings of IEEE International Conference on Machine Learning Applications*, 2011.

[14] K. Subramanian, C. Isbell, and A. Thomaz. Learning options through human interaction. In *2011 IJCAI Workshop on Agents Learning Interactively from Human Teachers (ALIHT)*, 2011.

[15] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[16] B. Tanner and A. White. RL-Glue: Language-independent software for reinforcement-learning experiments. *JMLR*, 10, 2009.

[17] M. Taylor, H. Suay, and S. Chernova. Integrating reinforcement learning with human demonstrations of varying ability. *AAMAS*, 2011.

[18] A. Tenorio-Gonzalez, E. Morales, and L. Villaseñor-Pineda. Dynamic reward shaping: training a robot by voice. *Advances in Artificial Intelligence–IBERAMIA 2010*, pages 483–492, 2010.

[19] A. Thomaz and C. Breazeal. Reinforcement Learning with Human Teachers: Evidence of Feedback and Guidance with Implications for Learning Performance. *AAAI*, 2006.

[20] E. Wiewiora, G. Cottrell, and C. Elkan. Principled methods for advising reinforcement learning agents. *ICML*, 2003.

Automatic Task Decomposition and State Abstraction from Demonstration

Luis C. Cobo
College of Engineering
Georgia Tech
Atlanta, GA 30332
luisca@gatech.edu

Charles L. Isbell Jr.
College of Computing
Georgia Tech
Atlanta, GA 30332
isbell@cc.gatech.edu

Andrea L. Thomaz
College of Computing
Georgia Tech
Atlanta, GA 30332
athomaz@cc.gatech.edu

ABSTRACT

Both Learning from Demonstration (LfD) and Reinforcement Learning (RL) are popular approaches for building decision-making agents. LfD applies supervised learning to a set of human demonstrations to infer and imitate the human policy, while RL uses only a reward signal and exploration to find an optimal policy. For complex tasks both of these techniques may be ineffective. LfD may require many more demonstrations than it is feasible to obtain, and RL can take an inadmissible amount of time to converge.

We present Automatic Decomposition and Abstraction from demonstration (ADA), an algorithm that uses mutual information measures over a set of human demonstrations to decompose a sequential decision process into several subtasks, finding state abstractions for each one of these subtasks. ADA then projects the human demonstrations into the abstracted state space to build a policy. This policy can later be improved using RL algorithms to surpass the performance of the human teacher. We find empirically that ADA can find satisficing policies for problems that are too complex to be solved with traditional LfD and RL algorithms. In particular, we show that we can use mutual information across state features to leverage human demonstrations to reduce the effects of the curse of dimensionality by finding subtasks and abstractions in sequential decision processes.

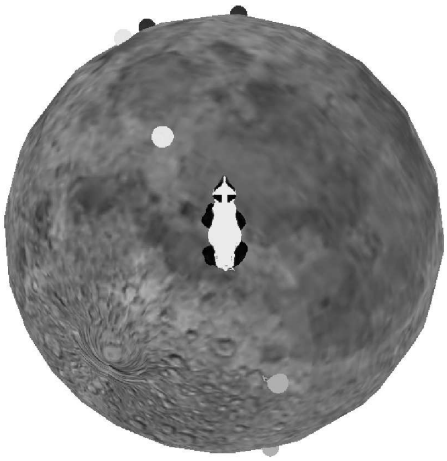


Figure 1: RainbowPanda domain. The agent must pick all balls, aiming at those that match the agent color at each moment. With 12 continuous state features, traditional RL does not converge in a reasonable amount of time. ADA finds a satisficing policy quickly by decomposing the problem into simpler subtasks.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Human Factors.

Keywords

Reinforcement learning, learning from demonstration, task decomposition, state abstraction.

1. INTRODUCTION

As it is impractical to implement manually every possible skill an agent might need to flourish in a human environment, our research aims to enable autonomous agents to learn new skills from *non-expert* humans without intervention from an engineer or programer. Agents such as non-player characters in video games or robotic domestic assistants must be able to adapt their behavior and learn new skills from everyday people.

Our approach derives a great deal from Learning from Demonstration (LfD) [3]. In the typical LfD scenario, an agent learns a mapping from states to actions using supervised learning techniques on a set of human demonstrations of a task. Such approaches assume that human demonstrations are positive examples of a near-optimal policy.

While LfD has proven successful [2], human demonstrations are expensive to obtain and, depending on the size of the state-action space, may require an impractical number of demonstrations. Additionally, small errors in imitating the human policy can compound and lead the agent to an unknown region of the state space. This large difference between training and testing state distributions can significantly impact performance.

In order to sidestep these shortcomings, we look for other kinds of information that human demonstrations can communicate to a learning agent. Instead of direct policy in-

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

formation, we use human demonstrations to learn a state abstraction and task decomposition to improve learning.

1.1 State Abstraction from Demonstration

We build on recent work in Abstraction from Demonstration [4] (AfD). In AfD, an agent uses human demonstrations on a task to figure which features of the state space are relevant for the task by measuring mutual information between each feature and the actions taken by the human teacher. Once the relevant features are identified, the agent builds an abstract state space in which the human policy can be expressed compactly, and then applies Reinforcement Learning (RL) algorithms to learn a policy in that abstract state space. If the original state space contains policy-invariant [9] features that can be ignored, this can lead to exponential speedups in skill learning, compared with traditional, raw-state space RL.

1.2 Task Decomposition from Abstraction

It is often the case that multipurpose agents have a high number of input signals of which only a subset are relevant for any specific task; however, using AfD does not help if all state features are relevant for the skill to be learned.

Our key insight is that there are often tasks where the entire state space is relevant for some part of the task, but the task can decomposed in subtasks so that for any given subtask there does exist an abstraction in which the policy can be expressed. For example, when we drive a car, we focus our attention almost completely on the car keys at the start and end of a drive, but completely ignore them for the rest of the trip. Our brain can receive simultaneously up to 11 million pieces of information, but it is estimated that at any given moment a person can be consciously aware of at most 40 of these [14].

Our goal is to infer this attentional focus, the particular task decomposition and state abstraction that the human demonstrator is using during their demonstration. We define a subtask as a region of the state space where only a subset of features is relevant, this subset being different from those of other subtasks. Thus we want a decomposition that maximizes our ability to apply AfD in each part.

1.3 Automatic Decomposition and Abstraction

In this paper we introduce *Automatic Task Decomposition and State Abstraction from Demonstration* (or Automatic Decomposition and Abstraction (ADA)), which uses human demonstrations to both decompose a skill into its subtasks and find independent state abstractions for each subtask. ADA can build more powerful abstractions than AfD, finding compact state space representations for more complex skills in which all state features are relevant at some point in time.

To determine which features are relevant to a particular subtask, we measure the mutual information between each feature of the state and the action taken by a human in a set of demonstrations. Once the state space is decomposed in different subtasks, the agent can learn and represent a compact policy by focusing only on the features that are relevant at each moment.

Fig.1 shows a simple example in our experimental domain. In this domain, an agent represented as a panda bear moves in an spherical surface. The agent can move forward and backwards and turn left and right. The overall task of the agent is to pick up all the balls, but at each moment it can only pick up balls of a specific color. With six balls, there are 12 continuous variables (relative angle and distance of each ball) and 1 discrete variable, the color the agent is currently allowed to pick up. In this 13-dimensional state space, traditional tabular RL takes an unreasonable amount of time to converge. Further, the complexity of the policy grows exponentially with the number of balls. As we shall see, with ADA, we can automatically decompose this problem into a set of subtasks, one per color, with each one needing to pay attention only to the closest ball of the target color. These 2-dimensional policies are easy to obtain, and the complexity of the global policy grows linearly with the number of balls.

After further situating our work in the next section, we describe in detail the ADA and ADA+RL algorithms and show that they can obtain good policies in problems where traditional RL and LfD algorithms offer poor performance.

2. RELATED WORK

Our work is at the intersection of different research lines, namely Learning from Demonstration, task decomposition, and state abstraction for RL.

LfD is a broad area of research, and several works explore how to combine demonstrations with traditional RL methods. Among these, using demonstrations or feedback to guide exploration [17, 12] or to learn a reward function [1] are complementary and could be combined with the method we propose. Other previous work uses demonstrations to extract task decompositions, like our method, but require a dynamic Bayesian network representation of the transitions and rewards models [16, 19], while our approach is model free.

There are also many approaches to task decomposition, but they usually require the user or the designer to explicitly specify the task structure [6, 10]. Others rely on heuristics that are adequate only for a very specific class of domains [8, 5, 18]. ADA is automatic and more general than these methods.

Regarding state abstraction in RL, prior work has used L_1 [13] and L_2 regularization [7], as well as selection from features that are based on Bellman error analysis [11]. While these approaches select features to represent a near-optimal value function, our work focuses on representing compactly a satisficing human policy, which is likely to be simpler and easier to learn than the optimal one. In the hierarchy for MDP state abstractions [15], ADA abstractions are in between a^*-irrelevance and π^*-irrelevance.

3. AUTOMATIC DECOMPOSITION AND ABSTRACTION

3.1 Preliminaries

We focus on sequential decision problems that can be expressed as Markov Decision Processes:

$$M = (S, A, P^a_{ss'}, R^a_s, \gamma)\,,$$

where S is a finite state space, A a finite set of actions, $P^a_{ss'} = Pr\,(s'|s,a)$ is the transition model, $R^a_s = r(s,a)$ the reward function and $0 \leq \gamma \leq 1$ the discount factor. $F = \{F_1, \ldots, F_n\}$ is the set of features of the state space, so that

$S = \{F_1 \times \cdots \times F_n\}$ and a state $s \in S$ is an n-tuple of features $s = (f_1, f_2, \ldots, f_n)$.

Solving an MDP means finding a policy $\pi : S \to A$ mapping states to actions. The value or sum of discount rewards of taking action a in state s and then following a policy π is

$$Q^\pi(s,a) = R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a Q^\pi(s', \pi(s')).$$

Most RL algorithms look for an optimal policy, *i.e.,* the policy that maximizes the sum of discounted reward,

$$\pi^*(s) = \arg\max_{a \in A} R_s^a + \gamma \sum_{s' \in S} P_{ss'}^a Q^{\pi^*}(s', \pi^*(s'));$$

however, ADA will aim to find a *satisficing policy*, *i.e.,* a policy that is comparable in performance to that of the human teachers.

We utilize usual notation for set operations, including $|\cdot|$ for cardinality. We use $\|\cdot\|$ for the L_2 norm of a vector.

Human demonstrations are defined as a set of episodes, each one comprising a list of state action pairs

$$H = \{\{(s_1, a_1), (s_2, a_2), \ldots\}, \ldots\}, s_i \in S, a_i \in A.$$

Mutual information is a measure of the amount of *entropy* in one random variable that can be explained by the value of a different random variable,

$$I(X;Y) = H(X) - H(X|Y) = H(X) + H(Y) - H(X,Y),$$

and can be computed as

$$I(X;Y) = \sum_{y \in Y} \sum_{x \in X} p(x,y) \log \left(\frac{p(x,y)}{p_x(x) p_y(y)} \right), \qquad (1)$$

where $p(x,y)$ is the joint *probability density function* (pdf) of random variables X and Y and $p_x(x)$ and $p_y(x)$ the respective marginal pdfs of each random variable. Thus, we define:

$$\vec{mi}_E = (I(F_1; A), \ldots, I(F_n; A))$$

as a vector whose elements are the mutual information between each feature of the state space and the action taken by the human teacher, according to the samples of H in the region $E \subset S$. To compute $\vec{mi}_E$, we estimate the appropriate joint and marginal pdfs using the samples of H that fall in region E, and use Equation 1.

3.2 Overview

Given an MDP M and a set of human demonstrations H for a skill to be learned, ADA finds a policy in three conceptual steps and an optional fourth step:

1. **Problem decomposition.** Using H, partition the state space S in different subtasks $T = \{t_1, t_2, \ldots\}$, $\cup T = S$, $t_i \cap t_j = \emptyset$ if $i \neq j$.
2. **Subtask state abstraction.** Using H, determine, for each subtask $t_i \in T$ the relevant features $\hat{F}_i = \{F_{i_1}, F_{i_2}, \ldots\}, F_{i_j} \in F$. and build a projection from the original state space S to the abstract state space $\phi(s) = \{i, f_{i_1,s}, f_{i_2,s}, \ldots\} \in \hat{S}, s \in t_i$.
3. **Policy construction.** Build a stochastic policy $\pi(\hat{s})$.
4. **Policy improvement.** Use Reinforcement Learning to improve over the policy found in the previous step. We refer to our algorithm as ADA+RL when it includes this step.

For clarity, we list the first two steps as if they are sequential; however, they are interwoven and concurrent. The decomposition of the state space depends on the quality of the abstractions that can be found on different subspaces.

3.3 Problem decomposition

Definition 1. A set of **subtasks** $T = \{t_1, t_2, \ldots\}$ of an MDP M are a set of regions of the state space S such that:

- The set of all subtasks T form a partition of the original state space S, *i.e.,* $\cup T = S$ and $t_i \cap t_j = \emptyset$ if $i \neq j$.
- A subtask t_i is identified by having a local satisficing policy π_i that depends only on a subset of the available features. This subset is different from neighboring subtasks.
- The global policy $\pi(\hat{s})$, the combination of the policies of each subtask, is also satisficing.

While this definition is not the typical one for subtasks in a sequential decision problem, it turns out to be a useful one, particularly if we focus on human-like activities. For example, cooking an elaborate recipe requires multiple steps, and each of these steps will involve different ingredients and utensils. It is possible that two conceptually different subtasks may depend on the same features, but in our framework, and arguably in general, the computational benefits of separating them are not significant.

With ADA, we can identify these subtasks given a set of human demonstrations H with two requirements:

- There must be a sufficient number of samples min_{ss} from each subtask in the set of demonstrations H.
- The class of possible boundaries between subtasks $B = \{b_1, b_2, , \ldots\}, b_i \subset S$ must be defined. Each boundary divides the state space in two, b_i and $S - b_i$. ADA will be able to find subtasks that can be expressed as combinations of these boundaries.

The necessary number of samples is determined in the first step of the ADA algorithm. This minimum sample size is needed due to the metric we use to infer feature relevance. Mutual information is sensitive to the *limited sampling bias*, and will be overestimated if the number of samples considered is too low.

The decomposition algorithm is described in Algorithm 1. At each iteration of the while loop, we consider a subspace E, with $E = S$ in the first iteration. We then consider all valid boundaries. If there are none, then E itself is a subtask. If there are valid boundaries, we score them and choose the one with the highest score. We then split E according to the boundary, and add the two new subspaces to the list of state spaces to be evaluated, to be further decomposed if necessary.

The boundaries B can have any form that is useful for the domain. In our experiments we consider thresholds on features, *i.e.,* axis-aligned surfaces. Using these boundaries,

Algorithm 1 ADA problem decomposition.

Require: MDP $M = (S, A, P^a_{ss'}, R^a_s, \gamma)$, $S = \{F_1 \times \ldots \times F_n\}$, human demonstrations $H = \{\{(s_1, a_1), (s_2, a_2), \ldots\}, \ldots\}, s \in S, a \in A$, boundaries $B = \{b_1, b_2, \ldots\}, b_i \in S$, ϵ.
$min_{ss} \leftarrow min\_sample\_size(H, \epsilon)$
$T \leftarrow \{\}$
$\mathbb{S} \leftarrow \{S\}$
while $\mathbb{S} \neq \emptyset$ **do**
  {pop removes the element from $\mathbb{S}$}
  $E \leftarrow \mathbb{S}.pop()$
  $B_E \leftarrow \{b \in B, valid\_split(b, E, min_{ss})\}$
  if $B_E = \emptyset$ **then**
    $T.push(E)$
  else
    $b_{best} \leftarrow \arg\max_{b \in B_E}(boundary\_score(b, E)))$
    $\mathbb{S}.push(b_{best} \cap E)$
    $\mathbb{S}.push((S - b_{best}) \cap E)$
  end if
end while
Return T

Algorithm 1 is just building a decision tree with a special split scoring function and stopping criteria.

In the next subsections we discuss the details of the split scoring function, the discriminator of valid boundaries, and the estimator of the minimum number of samples necessary. These contain the most interesting insights of ADA.

3.3.1 Boundary discriminator

Given a subspace $E \subset S$, m_{ss} and H, we consider a boundary $b \subset S$ to be valid if it meets three conditions:

1. There are enough samples in the set of human demonstration H to ensure we can measure mutual information with accuracy on both sides of the boundary, i.e.

$$|\{\{s, a\} \in H, s \in b \cap E\}| > min_{ss},$$
$$|\{\{s, a\} \in H, s \in (S - b) \cap E\}| > min_{ss}.$$

2. At least in one side of the boundary, either $b \cap E$ or $(S-b) \cap E$, it is possible to find a state abstraction, *i.e.,* some features are policy-invariant and can be ignored. We detail how we find these features in Section 3.4.

3. The state abstraction at both sides of the boundary is not the same.

This boundary discriminator works as the stopping criteria of the algorithm. When there are no more valid boundaries to be found, the decomposition step finishes.

3.3.2 Boundary scoring

The boundary scoring function determines the quality of b as a boundary between different subtasks within a region $E \in S$.

$$boundary\_score(b, E) = \left|\left| \frac{\vec{mi}_{b \cap E}}{\|\vec{mi}_{b \cap E}\|} - \frac{\vec{mi}_{(S-b) \cap E}}{\left|\left|\vec{mi}_{(S-b) \cap E}\right|\right|} \right|\right|. \quad (2)$$

The score is thus the euclidean distance between the normalized mutual information vectors at both sides of the boundary.

We are therefore measuring the difference between the relative importance of each feature at both sides of the boundary. As we want to find subtasks that rely on different features, we choose the boundary that maximizes this difference (see Algorithm 1).

3.3.3 Minimum samples

Due to the *limited sampling bias*, mutual information is overestimated if it is measured in an insufficient number of samples. The minimum samples in ADA, given a set of demonstrations H and parameter $\epsilon \approx 0.1$ is

$$m_{ss} = \arg\min_n average\left(\frac{\vec{mi}_S - \vec{mi}_{S,n}}{\vec{mi}_S}\right) > \epsilon, \quad (3)$$

where $mi_{S,n}$ is the mutual information vector on the original state space S, taking only a subset of n randomly chosen samples from all the samples in H. The subtraction and division are element-wise and the average function takes the average of the values of the resulting vector. Because of the variability of mutual information, it is necessary to evaluate Equation 3 several times for each possible n, each time with a different and independently chosen set of samples. Because of the limited sampling bias, the difference between $\vec{mi}_S$ and $\vec{mi}_{S,n}$ will grow as n decreases, and a binary search can be used to find m_{ss} efficiently.

3.4 Subtask state abstraction

Given a region of the state space $E \subset S$, we consider $\vec{mi}_E$ in order to estimate policy-irrelevant features. Even if a feature is completely irrelevant for the policy in a region of the space, its mutual information with the action will not be zero due to the limited sampling bias. Therefore, in ADA we group the values of $\vec{mi}_E$ in two clusters separated by the largest gap among the sorted values of the vector. If the value difference between any two features in different clusters is larger than the distance within a cluster, we consider we found a good abstraction that discards the features in the lower value cluster.

Note that this step occurs concurrently with the previous one, since the decomposition step needs to know in which regions of the state space there are good abstractions. Once these steps complete, we can build the projection function from the original function state space S to the abstract state space $\phi(s) = \{i, f_{i_1}, f_{i_2}, \ldots\} \in \hat{S}, s \in t_i$.

3.5 Policy construction

Once the task decomposition and state abstraction are completed and we have the projection function $\phi(s)$, we use the demonstrations H to build a stochastic policy that satisfies

$$P(\pi(\hat{s}) = a_i) = \frac{\left|\{\{s, a_i\} \in H, \phi(s) = \hat{s^*}\}\right|}{\left|\{\{s, a\} \in H, \phi(s) = \hat{s^*}, a \in A\}\right|}, \quad (4)$$

where $\hat{s^*}$ equals to $\hat{s}$ if $|\{\{s, a\} \in H, \phi(s) = \hat{s}, a \in A\}\}| > 0$. Otherwise, $\hat{s^*}$ equals to the nearest neighbor of $\hat{s}$ for which the denominator in Eq. 4 is not zero.

To compute the policy we project the state of each sample of H into the abstracted space and make a normalized histogram of each action. This concludes the basic ADA algorithm.

3.6 Policy improvement

ADA+RL adds another step, policy improvement, in which we use Reinforcement Learning techniques to find the optimal policy that can be represented in the abstract state space $\hat{S}$. Unlike traditional LfD techniques, ADA was designed so that the resulting policy can be easily improved given additional experience. In this way, we can obtain a better policy than that of the human teacher.

Given the kind of abstraction that ADA performs, bootstrapping methods such as Sarsa or Q-learning are not guaranteed to converge in the abstract state space[15]. As such, we can use either Monte Carlo methods or direct policy search. In Section 4 we choose to use policy search because it is fast and performs well given the compact state space that ADA generates.

4. EXPERIMENTAL RESULTS

4.1 Domains

To test the ADA and ADA+RL algorithms, we used two different domains, PandaSequential and RainbowPanda. We implemented a 3D game interface, shown in Figure 1, to capture human demonstrations. In both games, an agent (a panda bear) runs on a spherical surface collecting a series of colored balls. In PandaSequential, the agent must pick balls of different colors in a specific, fixed order, while in RainbowPanda, the agent is tinted with the color of the balls it is allowed to pick at any given moment. The color of the agent in RainbowPanda changes when the agent picks the last ball that matches the current color and may also change, with a small fixed probability, at any time step. With both domains, the initial position of the balls is assigned randomly at the start of each episode.

The state features are the distance and angle to each ball, relative to the agent. In RainbowPanda there are two balls of each color, so there are separate features for the closest ball and further ball from the agent. Distance and angle are measured in radians, and when a ball is not present (it has already been picked up), both features take a value outside of their normal range. RainbowPanda also has a discrete feature that contains the color of the agent, which is the color of the balls that the agent is allowed to pick up.

On both domains, the actions are move forward, backward, rotate right, rotate left, and no operation. The agents move backwards at a fourth of the speed they can move forward. The balls are picked up just by touching them. If the touched ball is the correct one to pick, the agent receives a positive reward and the ball disappears. Nothing happens if the agent touches a different ball. To compute the discounted reward, we use a discount factor $\gamma < 1$. The games were played at 20 frames per second, and this was also the rate at which the state was updated in the screen and an action was taken.

Both domains have similar interfaces, but their subtask structure is quite different. In PandaSequential, the subtasks have a fixed order, while in RainbowPanda there is no fixed order and every subtask may appear many times in the same episode. Additionally, in the first domain the current subtask is determined by the presence or absence of the balls (continuous variables) while in the second domain the current subtask is encoded in the color of the agent (a discrete variable).

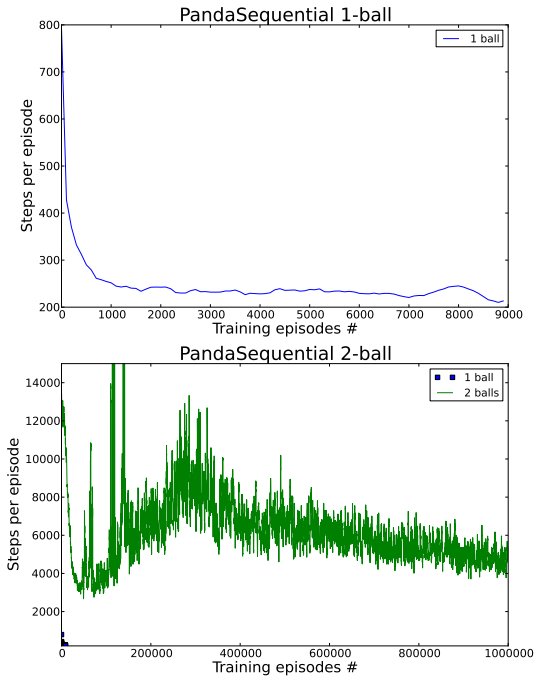


Figure 2: Results using Sarsa(λ) algorithm on a simplified version of the domain with only one or two balls. 1 extra ball decreases performance 2 orders of magnitude, even if the algorithm is left training 2k times longer.

To compare LfD, AfD and ADA, we captured a set of human demonstrations. We obtained 400 episodes for the PandaSequential domain (about 2 hours of gameplay) and 200 and 300 episodes, from different individuals, for the RainbowPanda domain (1.5h and 2.25h, respectively).

4.2 Results

Using the domains described above, we compared ADA with Reinforcement Learning using Sarsa(λ), Learning from Demonstration using a C4.5 decision tree, and Abstraction from Demonstration. We first discuss RL and LfD, then our algorithm, and finally we compare the abstractions that our algorithm and AfD found.

4.3 Reinforcement Learning using Sarsa

For Sarsa, we discretized the continuous values in 64 bins. The results were very poor in these domains because the dimensionality of the problems is so high. The simpler domain, PandaSequential, still has 64^6 possible states, for a total of about 343 billion Q-values. To ensure our implementation of the algorithm was correct and find its limits, we tested it with simplified versions of the game, with only one and two balls. The results are shown in Figure 2. We can see that Sarsa performs reasonably well for the case of only one ball (4096 states), but no longer so for the two-ball game (16.8 million states), even if leaving the algorithm running for 8 days in a modern computer, this is, 2000 times

Table 1: LfD results over 10K episodes, using a C4.5 decision tree. Avg. steps computed only over successful episodes.

Domain/Player	Episodes	Success rate	Avg. steps
Sequential	100	0.28%	241.21
	200	0.44%	227.79
	400	0.82%	242.83
Rainbow/A	100	8.1%	1543.84
	200	2.76%	1650.37
Rainbow/B	100	0.27%	1518.70
	200	0.27%	1994.92
	300	0.73%	1901.51

Table 2: Comparison of human performance, ADA, and ADA+RL, measured in number of steps to task completion averaged over 10 thousand episodes.

Domain/Player	Episodes	Human	ADA	ADA+RL
Sequential	100	322.67	360.75	267.11
	200	318.71	360.11	
	400	311.30	335.92	
Rainbow/A	100	525.33	-	-
	200	504.71	626.27	466.59
Rainbow/B	100	540.03	596.46	
	200	536.43	593.45	
	300	533.56	583.34	

longer than it took for the 1-ball policy to converge. The policy performance keeps improving, but at an extremely slow rate. Therefore, Sarsa is not an effective option for these domains.

4.4 Learning from Demonstration using C4.5

To compare with the performance of traditional LfD techniques, we trained a C4.5 decision tree with the demonstrations captured from human players, in a purely supervised learning fashion. We can see in Table 1 that LfD also performed poorly. The best result we obtained, using all available demonstrations, was a policy that would reach the goal state in less than 3% of the episodes[1].

4.5 Automatic Decomposition and Abstraction from Demonstration

To use ADA in our domains, we discretized the continuous values in 64 bins (same as for RL/Sarsa), used $\epsilon = 0.1$ and considered as candidate boundaries every possible threshold on every feature of the domain. ADA was much more effective than the other methods on both domains, and led to near-optimal polices that succeeded on every episode. Table 2 shows that we obtained policies comparable to those of the human teacher. Note that even though the average number of steps is slightly higher than for the LfD policy in the Sequential domain, this is averaged over all episodes, while the number for LfD is only averaged over the small percentage of episodes that LfD is able to resolve.

The success of ADA, compared with LfD and Sarsa, is due to its finding the right decomposition of the domains. For both domains, the algorithm builds an abstraction that focuses only on the angle with respect to the agent of the next ball to be picked up. Which ball is the target ball depends on what balls are present for the Sequential domain, and on what is the current color the agent is targeting for the Rainbow domain. The algorithm was able to identify the right boundary on each domain. It was a surprise that only the angle, and not the distance to the ball, was necessary, but it is easy to see that a satisficing policy can be found using only the angle: rotate until the ball is in front of the agent and then go forward. In fact, this was what the human players were doing, except in the rare case where the ball to pick up was right behind the agent; since the agent moves faster forward than backward it was usually not worth moving backwards.

Only one case in Table 2 did not produce the abstraction described above. Rainbow/B-100 episodes did not find any abstraction. This was due to m_{ss} being higher than a third of the total number of samples, therefore it could not find any of the 3 subtasks, one per color and roughly of the same size, that were found in the other cases. We tested a lower value for ϵ and in that case the usual abstraction was found.

In the same table we can see results for $ADA + RL$, applying policy search on top of the policy found by ADA. The abstraction built by ADA may prevent boot-strapping algorithms such as Sarsa or Q-learning from converging, but with only 192 states in the abstraction and a good starting policy, we can use direct policy search methods. We could obtain good results by just iteratively changing the policy of each state and evaluating the effect in performance using roll-outs.

Notice that, using this additional policy improvement step, we can find policies that are better than those demonstrated by the human teachers. The policies found were better than those demonstrated in three ways. First, the preferred action for states that were rarely visited was sometimes incorrect in the ADA policy because there were not enough samples in the demonstrations. ADA+RL could find the best action for these uncommon states. Second, human players would make the agent turn to face the target ball and then move forward when the relative angle to the ball was less than 15 degrees. ADA+RL found it was more efficient to turn until the angle to the ball was less than 3 degrees and only then move forward. Third, ADA policies assign some probability to each action depending how often it is taken in the demonstrations for a particular state. ADA+RL can identify which actions were not appropriate for the state and never execute them even if they appear in the demonstrations, maybe because of distractions or errors from the teacher. In short, the policy found by ADA+RL was a *more precise* and *less noisy* version of the policy derived directly from the demonstrations.

4.6 Abstraction from Demonstration

Finally, we tried AfD in the domains, using the abstraction algorithm described in Section 3.4 for the whole state space. In the Sequential domain, AfD would identify as the only useful feature the position of the first ball. This abstraction leads to a policy that can find the first ball quickly but can only perform a random walk to find the other two balls. The large difference in mutual information between each ball position and the action is due to the fact that while

[1]It should not be a surprise that sometimes, with more samples, the number of average steps on successful episodes increases. This is due to the policy being able to deal with more difficult episodes (remember that the initial placement of the balls is random) that require more steps to complete.

the first ball is present, its position is significant for the policy; however, the second ball is significant for the policy only half of the time it is present, and the third ball only a third of the time it is present.

Regarding AfD for the Rainbow domain, because the active color at each moment is chosen at random, the mutual information measures between each ball relative position and the action are similar. In this case, AfD is able to identify the true relevant features, *i.e.,* the relative position to the closest ball of each color. Due to the nature of AfD abstraction, we could not use bootstrapping algorithms such as Sarsa, and $64^3 = 262144$ states are too many for our naive policy search, so we tried to obtain a policy using Monte Carlo methods. Unfortunately, these are known to be much slower to converge than Sarsa and, even after experimenting with various exploration parameters, we could not reach a policy better than a random walk.

We can thus conclude that for complex domains that can be decomposed in different subtasks, ADA can find policies better than those demonstrated by humans, while traditional LfD, RL and AfD cannot find policies significantly better than a random walk.

4.7 Discussion

There are several advantages of ADA over traditional LfD. ADA can obtain much better performance from a small set of samples, while LfD often needs more samples than it is practical to obtain. In fact, for our two domains, we did not have the resources to collect a number of demonstrations large enough to obtain reasonable performance with LfD. Additionally, even with an arbitrarily large number of demonstrations, it is likely that ADA+RL can obtain policies better than those demonstrated and thus beat LfD, whose policy performance is limited by the quality of the demonstrations used.

Regarding RL techniques, it may seem unfair to compare those with ADA, as RL does not use the human demonstrations that are necessary for ADA. Yet, in the domains considered, even if we account for the time and cost of acquiring the human demonstrations, ADA still outperforms RL. We have seen in Section 4 that even for the simpler version of the Sequential domain with 2 balls, we still do not have a reasonable policy after a week. Using ADA, just with half an hour of human demonstrations and less than ten minutes of computing, we obtain a satisficing policy.

ADA is successful in these domains because it can find different state abstractions for different regions of the state space. Abstraction for Demonstration (AfD), a previous technique combining RL and LfD, only finds a single abstraction for the whole domain, and therefore it does not help much in the domains considered since all balls are relevant at some point during the task. AfD could make a small difference in performance by ignoring the distance to the balls, but the complexity of an AfD policy would still grow exponentially with the number of balls in the domain, while the complexity of the ADA policy grows linearly with the number of balls in the domain.

Obviously ADA cannot help if there is no possible decomposition of the domain and every feature is important at every moment; however we conjecture that such complex policies are rare, especially among tasks that can be demonstrated by humans. Humans have a limited capacity of attention and accomplish complex tasks by dividing them in manageable pieces. Therefore, if we have a set of demonstrations of a complex task, it is likely there are task decompositions to be found.

One "unfair" advantage of ADA over the other methods is that we must provide it with a set B of candidate boundaries, which is after all a form of domain information. In principle we could chose as boundary every possible subset of S, but this would be computationally intractable, so we must explicitly choose the candidate boundaries. This is a small price to pay for the performance gains of the algorithm. As a default choice, axis-aligned boundaries, i.e., thresholds in a single feature, are a compact class that works well across a diverse range of domains. They would work for the Taxi domain, which is the typical example of task decomposition in RL, using as boundary whether the passenger has been picked up or not yet. If the features are learned from low-level sensing information using unsupervised feature learning techniques, it is likely that one of the generated features will provide adequate thresholds. Additionally, many learning algorithms have similar kinds of bias; *e.g.,* decision trees also consider only thresholds in a single feature, just like ADA in our experimental setup.

A real limitation of ADA is that it does not consider second-order mutual information relationships, and these can be relevant. For example we can imagine a domain where the action to take depends on whether two independent random variables have the same value. The mutual information between each variable and the action might be 0, but the mutual information between both variables and the action would account for all the entropy of the action. We have decided to use only first-order mutual information because we believe it is enough to obtain a good decomposition of a wide range of problems and because the number of samples needed to get an accurate estimate of higher-order relationships is much larger. However, if a large number of demonstrations is available, ADA can be easily extended to use these additional mutual information measures.

One additional advantage of ADA, is that it can be used as part of a larger system of **transfer learning**. Once an autonomous agent learns a new skill and the subtasks it decomposes to, the subtask policies can be useful for other skills that may be decomposed in a similar way. An agent might, *e.g.,* as part of the policy improvement step for a specific subtask, try policies of previously learned subtasks that have the same abstraction, maybe after comparing policies and determining that the subtasks are similar. Demonstrating the utility of ADA for transfer learning is an important area of future work.

5. CONCLUSIONS

We have introduced Automatic Task Decomposition and State Abstraction from demonstration (ADA), an algorithm that leverages a small set of human demonstrations to decompose a skill in different subtasks, find abstractions for each of these subtasks, and build a compact satisficing policy for the skill. We have shown experimentally that, with a small number of demonstrations, ADA can easily find a policy for problems that are intractable using traditional RL and LfD techniques. Furthermore, we have shown that, given the structure of the policy that ADA finds, it can be improved to obtain a policy that outperforms the human teachers.

With this work we show that mutual information can be

used to extract useful domain knowledge of a sequential decision process from a set of human demonstrations. In the future, we plan to build upon this technique and combine it with function approximation and transfer learning.

6. ACKNOWLEDGMENTS

This work is supported by the National Science Foundation under Grant No. 0812116 and Obra Social "la Caixa".

7. REFERENCES

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. *International Conference on Machine Learning*, page 1, 2004.

[2] R. Aler, O. Garcia, and J. Valls. Correcting and improving imitation models of humans for robosoccer agents. In *IEEE Congress on Evolutionary Computation*, volume 3, 2005.

[3] B. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[4] L. C. Cobo, Z. Peng, C. L. Isbell, and A. L. Thomaz. Automatic Abstraction from Demonstration. *International Joint Conference on Artificial Intelligence*, pages 1243–1248, 2011.

[5] O. Şimşek and A. G. Barto. Using relative novelty to identify useful temporal abstractions in reinforcement learning. *International Conference on Machine Learning*, page 95, 2004.

[6] T. Dietterich. The MAXQ method for hierarchical reinforcement learning. In *International Conference in Machine Learning*, pages 118–126, 1998.

[7] A. Farahmand, M. Ghavamzadeh, C. Szepesvari, and S. Mannor. Regularized policy iteration. *Advances in Neural Information Processing Systems*, 21:441–448, 2009.

[8] B. Hengst. Discovering hierarchy in reinforcement learning with HEXQ. In *International Conference on Machine Learning*, pages 234–250, 2002.

[9] N. K. Jong and P. Stone. State Abstraction Discovery from Irrelevant State Variables. *International Joint Conference on Artificial Intelligence*, (August):752–757, 2005.

[10] A. Jonsson and A. Barto. Automated state abstraction for options using the U-tree algorithm. *Advances in Neural Information Processing Systems*, pages 1054–1060, 2001.

[11] P. W. Keller, S. Mannor, and D. Precup. Automatic basis function construction for approximate dynamic programming and reinforcement learning. *International Conference on Machine Learning*, pages 449–456, 2006.

[12] W. Knox and P. Stone. Combining Manual Feedback with Subsequent MDP Reward Signals for Reinforcement Learning. *Annual International Conference on Autonomous Agents and Multiagent Systems*, pages 10–14, 2010.

[13] J. Z. Kolter and A. Y. Ng. Regularization and feature selection in least-squares temporal difference learning. *International Conference on Machine Learning*, 94305:1–8, 2009.

[14] J. Kristeva. *Strangers to ourselves*. European Perspectives: A Series In Social Thought And Cultural Criticism. Columbia University Press, 1991.

[15] L. Li, T. J. Walsh, and M. L. Littman. Towards a Unified Theory of State Abstraction for MDPs. In *International Symposium on Artificial Intelligence and Mathematics*, pages 531–539, 2006.

[16] N. Mehta, M. Wynkoop, S. Ray, P. Tadepalli, and T. Dietterich. Automatic induction of MAXQ hierarchies. In *NIPS Workshop: Hierarchical Organization of Behavior*, pages 1–5, 2007.

[17] W. Smart and L. Pack Kaelbling. Effective reinforcement learning for mobile robots. *IEEE International Conference on Robotics and Automation*, pages 3404–3410, 2002.

[18] M. Stolle and D. Precup. Learning Options in Reinforcement Learning. *Abstraction, Reformulation, and Approximation*, pages 212–223, 2002.

[19] P. Zang, P. Zhou, D. Minnen, and C. Isbell. Discovering options from example trajectories. *International Conference on Machine Learning*, pages 1217–1224, 2009.

Session 4C
Argumentation & Negotiation

Quantifying Disagreement in Argument-based Reasoning

Richard Booth
University of Luxembourg
Computer Science & Communication
richard.booth@uni.lu

Martin Caminada, Mikołaj Podlaszewski
University of Luxembourg
Security, Reliability & Trust
martin.caminada@uni.lu
mikolaj.podlaszewski@uni.lu

Iyad Rahwan
Masdar Institute of Science & Technology, UAE
Massachusetts Institute of Technology, USA
irahwan@acm.org

ABSTRACT

An argumentation framework can be seen as expressing, in an abstract way, the conflicting information of an underlying logical knowledge base. This conflicting information often allows for the presence of more than one possible reasonable position (extension/labelling) which one can take. A relevant question, therefore, is how much these positions differ from each other. In the current paper, we will examine the issue of how to define meaningful measures of distance between the (complete) labellings of a given argumentation framework. We provide concrete distance measures based on argument-wise label difference, as well as based on the notion of critical sets, and examine their properties.

Categories and Subject Descriptors

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

General Terms

Algorithms, Measurement, Theory

Keywords

argumentation, distance measures, complete labellings, aggregation, belief revision

1. INTRODUCTION

Given a conflicting logical theory, an agent is faced with the problem of deciding what it could reasonably believe. As advocated in various nonmonotonic inference formalisms such as default logic [24], it is often possible to identify *multiple* reasonable positions, or so-called *extensions*. This idea has been adopted in abstract argumentation theory [14], which attempts to analyze possible extensions while abstracting away from the underlying logic. In particular, this theory views logical derivations as abstract arguments (nodes in a graph), and conflicts as defeat relations (directed arcs) over these arguments.

The presence of multiple reasonable positions raises a fundamental question: *how different are two given evaluations of a conflicting logical theory?* We attempt to answer this question in the context of abstract argumentation theory.

This question is relevant to two fundamental problems. The first problem is *argument-based belief revision*. Suppose a diplomat receives instructions to switch his position on one particular argument (see Section 3 for an example). To maintain a consistent viewpoint, the diplomat must revise his evaluation of other related arguments. Faced with multiple possibilities, the diplomat may wish to choose the one that differs the least from his initial position (e.g. to maintain credibility).

The issue of distance is also relevant to the problem of *judgement aggregation* over how a given set of arguments should be evaluated collectively by a group of agents with different opinions [11, 12, 23]. For instance it is very well possible that the members of a jury in a criminal trial all share the same information on the case (and hence have the same argumentation framework) but still have different opinions on what the verdict should be. Hence, these differences of opinion are consequences not of differences in the knowledge base but of the nature of nonmonotonic reasoning, which allows for various reasonable positions (extensions). In the context of judgement aggregation one may examine the extent to which the collective position differs from the various positions of the individual participants. Ideally, one would like to have a collective position that is closest to the collection of individual positions, for example such that the sum of its distance to each individual position is minimal.

In this paper, we examine a number of possible candidates for measuring the *distance* between different labellings (evaluations) of an argumentation graph. The paper advances the state-of-the-art in argument-based reasoning in three ways: (1) We provide the first systematic investigation of quantifying the distance between two evaluations of an argument graph; (2) We examine a number of intuitive measures and show that they fail to satisfy basic desirable postulates; (3) we come up with a measure that satisfies them all. In addition to providing many answers, our paper also raises many interesting questions to the community at the intersection between argumentation and social choice.

2. ABSTRACT ARGUMENTATION

In this section, we briefly restate some preliminaries regarding argumentation theory. For simplicity, we only consider finite argumentation frameworks.

Definition 1. An *argumentation framework* (AF for short) is a pair $\mathcal{A} = (Ar, \rightharpoonup)$, where Ar is a finite set of arguments

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

and $\rightharpoonup \subseteq Ar \times Ar$.

We say that argument A *attacks* argument B iff $(A, B) \in \rightharpoonup$. An AF can be represented as a directed graph in which the arguments are represented as nodes and the attack relation is represented as arrows.

In the current paper, we follow the approach of [6, 10] in which the semantics of abstract argumentation is expressed in terms of *argument labellings*. The idea is to distinguish between the arguments that one accepts (that are labelled `in`), the arguments that one rejects (that are labelled `out`) and the arguments which one abstains from having an opinion about (that are labelled `undec` for "undecided").

Definition 2. Given an AF $\mathcal{A} = (Ar, \rightharpoonup)$, a labelling for $\mathcal{A}$ is a function $\mathcal{L} : Ar \rightarrow \{\texttt{in}, \texttt{out}, \texttt{undec}\}$.

Since a labelling is a function, it can be represented as a set of pairs, each consisting of an argument and a label (`in`, `out`, or `undec`). We are now ready to state the concept of *complete labelling* [6, 10].

Definition 3. Let $\mathcal{L}$ be a labelling for AF $\mathcal{A} = (Ar, \rightharpoonup)$. $\mathcal{L}$ is a complete labelling (over $\mathcal{A}$) iff for each $A \in Ar$ it holds that:

1. $\mathcal{L}(A) = \texttt{in}$ iff $\forall B \in Ar : (B \rightharpoonup A \supset \mathcal{L}(B) = \texttt{out})$
2. $\mathcal{L}(A) = \texttt{out}$ iff $\exists B \in Ar : (B \rightharpoonup A \wedge \mathcal{L}(B) = \texttt{in})$.

We denote the set of all complete labellings of $\mathcal{A}$ by $Comp_{\mathcal{A}}$.

As stated in [6, 10], complete labellings coincide with complete extensions in the sense of [14]. Moreover, the relationship between them is one-to-one. In essence, a complete extension is simply the `in`-labelled part of a complete labelling [6, 10].

The labelling approach has also been defined for other semantics, such as grounded, preferred, stable and semi-stable semantics, as well as for ideal semantics (see the overview article [2] for details). In this paper, however, we will focus on the case of complete semantics and the associated complete labellings, not only because of their relative simplicity, but also because complete labellings serve as the basis for defining labellings for various other semantics [10]. That is, semantics like grounded [14], preferred [14], stable [14], semi-stable [9], ideal [15] and eager [7] in essence select subsets of the set of all complete labellings (see [2]). Since the approach in the current paper is to compare any arbitrary pair of complete labellings, our results are directly applicable also to the aforementioned semantics.[1]

Example 1. Consider a simple argumentation framework $\mathcal{A} = (Ar, \rightharpoonup)$ with $Ar = \{A, B, C\}$ and $\rightharpoonup = \{(A, B), (B, A), (B, C)\}$. Then $Comp_{\mathcal{A}} = \{\mathcal{L}_1, \mathcal{L}_2, \mathcal{L}_3\}$, where each $\mathcal{L}_i$ may be visualised in Fig. 1. In this and subsequent diagrams, a node with a solid line indicates an `in` label, a dotted line indicates `out` and a grey node indicates `undec`. Thus for example the first labelling $\mathcal{L}_1 = \{(A, \texttt{in}), (B, \texttt{out}), (C, \texttt{in})\}$.

[1]Another point to mention is that it has been proved that complete-based semantics (that is, semantics whose sets of extensions/labellings are subsets of the set of all complete extensions/labellings), when used for the purpose of logical inference, tend to produce fully instantiated argumentation formalisms that satisfy reasonable properties in the sense of [8, 22].

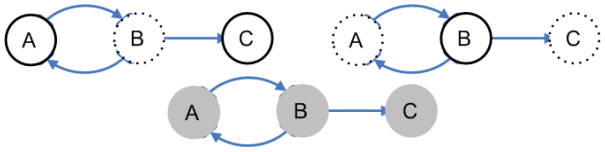


Figure 1: Three possible complete labellings $\mathcal{L}_1, \mathcal{L}_2$ and $\mathcal{L}_3$

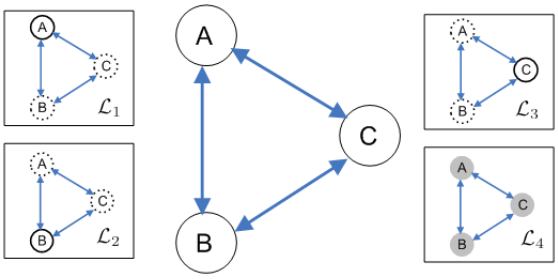


Figure 2: The AF $\mathcal{A}_1$ and its four possible complete labellings $\mathcal{L}_1$-$\mathcal{L}_4$

3. BELIEF REVISION AND JUDGEMENT AGGREGATION

Consider three arguments about global warming, each one grounded in some scientific evidence, with the following associated conclusions:

- A: Global warming is mainly caused by volcanic activity.
- B: Global warming is mainly caused by natural variation in solar radiation.
- C: Global warming is a human-induced phenomenon.

Clearly, it is not possible to subscribe to both arguments A and B, since they attribute global warming to different major causes. However, both these arguments attack argument C, which attributes global warming to human activity. This situation can be modelled with the AF $\mathcal{A}_1$ shown in Fig.2. Here there are four possible complete labellings $\mathcal{L}_1$-$\mathcal{L}_4$, which are also depicted.

Suppose a diplomat was initially adopting the position corresponding to labelling $\mathcal{L}_3$ in Fig. 2, which focusses on human activity (argument C) as the main cause of global warming.

However, recent elections in his home country gave rise to a new climate-sceptical government, which requires the diplomat to change his position in order to no longer accept argument C (that is: to revise his complete labelling to one in which C is no longer labelled `in`). He is now faced with two possible revisions of his original position $\mathcal{L}_3$. On one hand, he can switch to labelling $\mathcal{L}_1$ or $\mathcal{L}_2$, citing the alternative theory. On the other hand, he could switch to labelling $\mathcal{L}_4$, admitting that the matter cannot be decided.

Let us assume that politicians like to maintain a reputation of being generally consistent. Therefore, when they switch their points of view, they like to minimize the extent to which they deviate from their original positions. In the

above example, it is far from obvious which revision of the original position $\mathcal{L}_3$ is less dramatic. On one hand, switching to the alternative theories $\mathcal{L}_1$ or $\mathcal{L}_2$ keeps the status of at least one argument the same, while switching to labelling $\mathcal{L}_4$ requires changing the status of all arguments involved. On the other hand, switching the status of argument C from being fully accepted to being completely rejected (as in $\mathcal{L}_1$ or $\mathcal{L}_2$) seems more severe than simply moving to a position of indecision (as in $\mathcal{L}_4$).

Following up on the example above, suppose we have a panel consisting of three scientists, with two supporting position $\mathcal{L}_1$, and one supporting position $\mathcal{L}_2$. Suppose the scientists want to reach a collective position that is closest to their respective individual positions, in order to minimize the degree to which they individually deviate from their original positions. To achieve this, should all of them concede to the third undecided position $\mathcal{L}_4$, as is suggested in [11]? Or should the third scientist individually concede to position $\mathcal{L}_1$, ensuring the first two stick to their view? The answer to this question relies crucially on how we quantify the distance between the different positions.

The examples above highlight the need for a systematic approach to identifying the extent to which two positions differ, ideally creating a reliable quantitative measure of distance between different complete labellings.

4. DISTANCE BETWEEN LABELLINGS

The problem we are interested is the following:

> Given an AF $\mathcal{A}$, and given two complete labellings $\mathcal{S}$ (the *source* labelling) and $\mathcal{T}$ (the *target* labelling) over $\mathcal{A}$, how can we quantify the *distance* from $\mathcal{S}$ to $\mathcal{T}$, denoted $d(\mathcal{S}, \mathcal{T})$?

Of course we don't just want a method which applies to only one AF, we want a method to be able to do this for *any* given $\mathcal{A}$.

Definition 4. A *labelling distance* (for AF $\mathcal{A}$) is a function $d : \mathit{Comp}_{\mathcal{A}} \times \mathit{Comp}_{\mathcal{A}} \to \mathbb{N}$. A *labelling distance method* is a function which assigns to every AF $\mathcal{A}$ a labelling distance for $\mathcal{A}$.

In the following sections we will provide a few concrete definitions of distance functions. But first, are there any properties which we should expect such a function to satisfy?

4.1 Properties for distance methods

In mathematics, when formalising the notion of distance it is common to require that d be a *metric*. In our present setting that means that the following hold for all complete labellings $\mathcal{S}, \mathcal{T}, \mathcal{U}$ over a given AF $\mathcal{A}$:

(dm1) $d(\mathcal{S}, \mathcal{S}) = 0$
(dm2) $d(\mathcal{S}, \mathcal{T}) > 0$ if $\mathcal{S} \neq \mathcal{T}$
(dm3) $d(\mathcal{S}, \mathcal{T}) = d(\mathcal{T}, \mathcal{S})$ (Symmetry)
(dm4) $d(\mathcal{S}, \mathcal{T}) \leq d(\mathcal{S}, \mathcal{U}) + d(\mathcal{U}, \mathcal{T})$ (Triangle inequality)

Also, let's define the following binary relation over $\mathit{Comp}_{\mathcal{A}}$, given a fixed source complete labelling $\mathcal{S}$:

$$\mathcal{T}_1 \leq_{\mathcal{S}} \mathcal{T}_2 \text{ iff } \forall A \, (\mathcal{T}_1(A) = \mathcal{S}(A) \vee \mathcal{T}_2(A) = \mathcal{T}_1(A))$$

$\mathcal{T}_1 \leq_{\mathcal{S}} \mathcal{T}_2$ means that every argument that $\mathcal{T}_1$ labels differently from $\mathcal{S}$, is labelled equally differently by $\mathcal{T}_2$. Thus $\mathcal{T}_2$ differs from $\mathcal{S}$ at least as much as $\mathcal{T}_1$ does. It can be shown that $\leq_{\mathcal{S}}$ is a partial order over $\mathit{Comp}_{\mathcal{A}}$ with minimum element $\mathcal{S}$, i.e., $\mathcal{S} \leq_{\mathcal{S}} \mathcal{T}$ for all $\mathcal{T} \in \mathit{Comp}_{\mathcal{A}}$. Let $<_{\mathcal{S}}$ denote the strict version of $\leq_{\mathcal{S}}$, i.e., $\mathcal{T}_1 <_{\mathcal{S}} \mathcal{T}_2$ iff both $\mathcal{T}_1 \leq_{\mathcal{S}} \mathcal{T}_2$ and $\mathcal{T}_2 \not\leq_{\mathcal{S}} \mathcal{T}_1$. Thus the following might seem to be a reasonable requirement on a distance function d:

(dm5) If $\mathcal{T}_1 <_{\mathcal{S}} \mathcal{T}_2$ then $d(\mathcal{S}, \mathcal{T}_1) < d(\mathcal{S}, \mathcal{T}_2)$ (Disagreement monotonicity)

To see why this might be reasonable, note that $\mathcal{T}_1 <_{\mathcal{S}} \mathcal{T}_2$ means that for every argument on which $\mathcal{T}_1$ disagrees with $\mathcal{S}$, the labelling $\mathcal{T}_2$ disagrees with $\mathcal{S}$ in exactly the same way, but that there exists at least one argument on which $\mathcal{T}_2$ disagrees with $\mathcal{S}$, but for which $\mathcal{T}_1$ and $\mathcal{S}$ agree. In this case it seems as though $\mathcal{T}_2$ is making strictly more changes to $\mathcal{S}$ than $\mathcal{T}_1$ is, and so d should also endorse this conclusion. It is not difficult to show that if d satisfies both **(dm1)** and **(dm5)** then it satisfies **(dm2)**.

We can also describe a postulate which is stronger than **(dm5)**. To express this property we first define the following ordering over $\mathit{Comp}_{\mathcal{A}}$, given any source labelling $\mathcal{S}$ and target labellings $\mathcal{T}_1, \mathcal{T}_2$:

$$\mathcal{T}_1 \leq^b_{\mathcal{S}} \mathcal{T}_2 \text{ iff } \forall A \left(\begin{array}{l} \mathcal{T}_1(A) = \mathcal{S}(A) \vee \mathcal{T}_1(A) = \mathcal{T}_2(A) \\ \vee [\mathcal{T}_1(A) = \texttt{undec} \wedge \mathcal{S}(A) \neq \mathcal{T}_2(A)] \end{array} \right)$$

Like $\leq_{\mathcal{S}}$, the ordering $\leq^b_{\mathcal{S}}$ forms a partial order with minimum element $\mathcal{S}$. The superscript "b" on $\leq^b_{\mathcal{S}}$ may be thought of as standing for "*between*", since $\mathcal{T}_1 \leq^b_{\mathcal{S}} \mathcal{T}_2$ is merely expressing that, for all $A \in Ar$, $\mathcal{T}_1(A)$ lies on a path *between* $\mathcal{S}(A)$ and $\mathcal{T}_2(A)$, assuming the neighbourhood graph $\texttt{in} - \texttt{undec} - \texttt{out}$ over the labels. We may then propose the following:

(dm5+) If $\mathcal{T}_1 <^b_{\mathcal{S}} \mathcal{T}_2$ then $d(\mathcal{S}, \mathcal{T}_1) < d(\mathcal{S}, \mathcal{T}_2)$ (Betweenness monotonicity)

where $<^b_{\mathcal{S}}$ is the strict part of the relation $\leq^b_{\mathcal{S}}$. Since clearly $\mathcal{T}_1 <_{\mathcal{S}} \mathcal{T}_2$ implies $\mathcal{T}_1 <^b_{\mathcal{S}} \mathcal{T}_2$ we have that **(dm5+)** is indeed a strengthening of **(dm5)**.

5. SUM-BASED DISTANCE

Our first family of distance functions is about simply finding the raw quantity of disagreement between two complete labellings. We can do this in terms of difference between labels, that is, we assume we have some measure of disagreement $\mathit{diff}(\mathtt{x}, \mathtt{y})$ for any $\mathtt{x}, \mathtt{y} \in \{\texttt{in}, \texttt{out}, \texttt{undec}\}$ between the different labels, and then obtain the distance between two labellings by summing the differences between *all* arguments in the AF $(Ar, \rightharpoonup)$ under consideration, i.e., take

$$d(\mathcal{S}, \mathcal{T}) = \sum_{A \in Ar} \mathit{diff}(\mathcal{S}(A), \mathcal{T}(A)). \quad (1)$$

Definition 5. If the function d can be defined from some function $\mathit{diff} : \{\texttt{in}, \texttt{out}, \texttt{undec}\}^2 \to \mathbb{N}$ as in (1) then we say d is a *simple diff-based distance method*.

It turns out that the results in this paper depend only on a few fundamental requirements on *diff*, encapsulated in the following definition:

Definition 6. A *basic label difference measure* is a function $\mathit{diff} : \{\texttt{in}, \texttt{out}, \texttt{undec}\}^2 \to \mathbb{N}$ which satisfies the following properties, for all $\mathtt{x}, \mathtt{y} \in \{\texttt{in}, \texttt{out}, \texttt{undec}\}$:

(diff 1) $\mathit{diff}(\mathtt{x},\mathtt{x}) = 0$
(diff 2) $\mathit{diff}(\mathtt{x},\mathtt{y}) = \mathit{diff}(\mathtt{y},\mathtt{x})$
(diff 3) $\mathit{diff}(\mathtt{in},\mathtt{out}) > 0$
(diff 4) $\mathit{diff}(\mathtt{in},\mathtt{undec}) = \mathit{diff}(\mathtt{out},\mathtt{undec})$

Note that this means, in effect, any simple diff-based measure based on a basic label difference measure is completely specified by 2 quantities: $\mathit{diff}(\mathtt{in},\mathtt{undec})$ and $\mathit{diff}(\mathtt{in},\mathtt{out})$, which may respectively be thought of as the costs attached to a *soft* and *hard* conflict. From now on any unspecified *diff*-measure will be assumed to satisfy **(diff 1)**-**(diff 4)**.

PROPOSITION 1. *If d is a simple diff-based distance method defined via a basic label difference measure then d satisfies* **(dm1)** *and* **(dm3)**.

(We remark that Propositions 1-4 in this section actually all follow as corollaries of a more general result, Theorem 1, in Section 6.1). As we will see below, the remaining distance properties from the previous section can easily be captured by placing further, optional, constraints on *diff*. Let us take in a few concrete examples.

Measuring incompatibility

The property **(diff 3)** ensures that a hard conflict always contributes a strictly positive value. But note we do not require a soft conflict to do the same. That is, we do not insist on the following strengthening of **(diff 3)**:

(diff 3+) $\mathit{diff}(\mathtt{x},\mathtt{y}) > 0$ for $\mathtt{x} \neq \mathtt{y}$

In this way we allow *diff*-measures such as the following, which is inspired by the work of [11]. A labelling $\mathcal{L}_1$ is *compatible* with labelling $\mathcal{L}_2$ (written as $\mathcal{L}_1 \approx \mathcal{L}_2$) iff there is no argument A such that either $[\mathcal{L}_1(A) = \mathtt{in}$ and $\mathcal{L}_2(A) = \mathtt{out}]$ or $[\mathcal{L}_1(A) = \mathtt{out}$ and $\mathcal{L}_2(A) = \mathtt{in}]$. The idea behind compatibility is to give a rough impression of how difficult it is to publicly defend a position (labelling) that is not one's own. Although it might be possible to publicly accept or reject an argument which one privately has no opinion about (undec), or to remain silent about an argument that one privately accepts or rejects, it is significantly more difficult to publicly accept an argument which one privately rejects (and vice versa). Our first concrete measure of distance makes the distance zero if the two labellings are compatible, and measures the "degree of incompatibility" if they are not.

$$\mathit{diff}^{\approx}(\mathtt{in},\mathtt{out}) = 1, \qquad \mathit{diff}^{\approx}(\mathtt{in},\mathtt{undec}) = 0.$$

This leads to a function $d^{\approx}$ (defined using $\mathit{diff}^{\approx}$ via (1)) which is more like a "measure of conflict" between $\mathcal{S}$ and $\mathcal{T}$. Measure $d^{\approx}$ fails to satisfy **(dm2)**, as can be seen in Fig. 1, where $d^{\approx}(\mathcal{L}_1,\mathcal{L}_3) = 0$. If we *do* insist on **(diff 3+)** then we ensure not only **(dm2)** but also **(dm5)**:

PROPOSITION 2. *If d is a simple diff-based distance method defined via a basic label difference measure which satisfies* **(diff 3+)** *then d satisfies* **(dm5)** *(and hence also* **(dm2)***).*

Hamming distance

A very simple example of a *diff*-measure satisfying **(diff 3+)** is as follows:

$$\mathit{diff}^{H}(\mathtt{in},\mathtt{out}) = \mathit{diff}^{H}(\mathtt{in},\mathtt{undec}) = 1.$$

Then the distance between $\mathcal{S}$ and $\mathcal{T}$ boils down to the number of arguments on which $\mathcal{S}$ and $\mathcal{T}$ differ, i.e., the Hamming distance between $\mathcal{S}$ and $\mathcal{T}$. Let d^H denote the distance defined using diff^H. Consider for instance the results for Fig. 1, where we see that $d^H(\mathcal{L}_1,\mathcal{L}_2) = 3 = d^H(\mathcal{L}_1,\mathcal{L}_3)$. Thus, according to d^H, labellings $\mathcal{L}_2$ and $\mathcal{L}_3$ are equidistant from $\mathcal{L}_1$. However it might be thought that the change between $\mathcal{L}_1$ and $\mathcal{L}_2$ is more "drastic" than that between $\mathcal{L}_1$ and $\mathcal{L}_3$, since it involves a complete swing in the status of its arguments from in (resp. out) to out (resp. in). This example demonstrates that d^H fails to satisfy **(dm5+)**, since $\mathcal{L}_3 <^b_{\mathcal{L}_1} \mathcal{L}_2$ but $d^H(\mathcal{L}_1,\mathcal{L}_3) \not< d^H(\mathcal{L}_1,\mathcal{L}_2)$. Shouldn't the difference between in and out be strictly greater than the difference between in (or out) to undec? In other words we might expect:

(diff 5) $\mathit{diff}(\mathtt{in},\mathtt{out}) > \mathit{diff}(\mathtt{in},\mathtt{undec})$

PROPOSITION 3. *If d is a simple diff-based distance method defined via a basic label difference measure which satisfies* **(diff 3+)** *and* **(diff 5)** *then d satisfies* **(dm5+)**.

Refined Hamming distance

An easy way to define a basic label difference measure which satisfies both **(diff 3+)** and **(diff 5)** is to set:

$$\mathit{diff}^{rh}(\mathtt{in},\mathtt{out}) = 2, \qquad \mathit{diff}^{rh}(\mathtt{in},\mathtt{undec}) = 1,$$

where rh stands for "refined Hamming". Note $\mathit{diff}^{rh}(\mathtt{x},\mathtt{y})$ may be thought of as the length of the shortest path between x and y in the neighbourhood graph $\mathtt{in} - \mathtt{undec} - \mathtt{out}$ over the labels. We denote by d^{rh} the distance obtained by plugging diff^{rh} into (1). Going back to Fig. 1, we have $d^{rh}(\mathcal{L}_1,\mathcal{L}_2) = 3 \times \mathit{diff}^{rh}(\mathtt{in},\mathtt{out}) = 6$ and $d^{rh}(\mathcal{L}_1,\mathcal{L}_3) = 3 \times \mathit{diff}^{rh}(\mathtt{in},\mathtt{undec}) = 3$, yielding the expected $d^{rh}(\mathcal{L}_1,\mathcal{L}_3) < d^{rh}(\mathcal{L}_1,\mathcal{L}_3)$. Propositions 1-3 aready tell us d^{rh} satisfies all the distance properties from the previous section. The only one which remains is the triangle inequality **(dm4)**. But in fact this too is satisfied, owing to the fact that diff^{rh} satisfies the following (which implies **(diff 3+)** for basic label difference measures):

(diff 3++) $2 \times \mathit{diff}(\mathtt{in},\mathtt{undec}) \geq \mathit{diff}(\mathtt{in},\mathtt{out})$

This property actually ensures that *diff* itself satisfies the triangle inequality over the set of labels.

PROPOSITION 4. *If d is a simple diff-based distance method defined via a basic label difference measure which satisfies* **(diff 3++)** *then d satisfies* **(dm4)**.

Since diff^H obviously satisfies **(diff 3++)** this means we also get that d^H satisfies **(dm4)**. The incompatibility distance $d^{\approx}$, however, does not satisfy **(dm4)**, as can be seen in Fig. 1, where $d^{\approx}(\mathcal{L}_1,\mathcal{L}_2) = 3 > 0 = d^{\approx}(\mathcal{L}_1,\mathcal{L}_3) + d^{\approx}(\mathcal{L}_3,\mathcal{L}_2)$.

6. CRITICAL SETS APPROACHES

Suppose we have the complete labelling $\mathcal{S}$ shown at the top of Fig. 3 over an AF containing eight arguments $\{A, B, C, D, E, F, G, H\}$. As usual a node with a solid line denotes the argument is in, while a dotted line denotes out. Now consider the two target labellings $\mathcal{T}_1$ and $\mathcal{T}_2$ shown below it. $\mathcal{T}_1$ is obtained from $\mathcal{S}$ by leaving the labels of A, B, C, D as they are and inverting the labels of the four arguments E, F, G, H. For $\mathcal{T}_2$ we leave E, F, G, H untouched and invert the labels of the four arguments A, B, C, D. The question is: which of $\mathcal{T}_1$, $\mathcal{T}_2$ is closer to $\mathcal{S}$? Or are they both equally close?

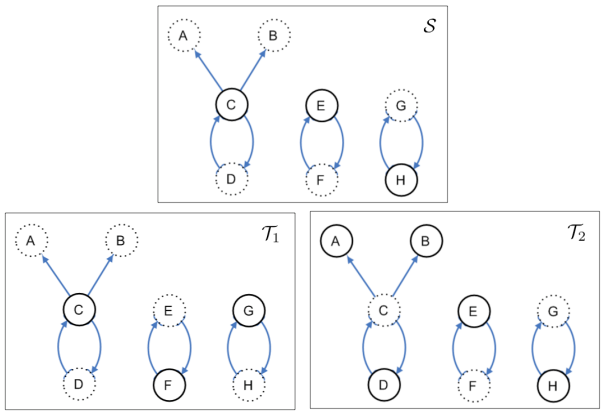


Figure 3: Source labelling $\mathcal{S}$ and 2 target labellings $\mathcal{T}_1$, $\mathcal{T}_2$

Let's consider what a simple diff-based distance function d has to say about this. One can see that we will get $d(\mathcal{S},\mathcal{T}_1) = 4 \times \mathit{diff}(\texttt{in},\texttt{out}) = d(\mathcal{S},\mathcal{T}_2)$. Thus *any* simple diff-based distance will judge $\mathcal{T}_1$ and $\mathcal{T}_2$ as equidistant from $\mathcal{S}$.

However, on reflection it seems we can be more reasonable and say that $\mathcal{T}_2$ is closer to $\mathcal{S}$. Intuitively the reason is based on the observation that disagreement between $\mathcal{S}$ and $\mathcal{T}_2$ involves a higher degree of "contagion". If two agents only differ in their opinions on argument C (or only on D), this would suffice to determine their disagreement over all other arguments in that connected component (namely A, B, C, and D). On the other hand, when comparing $\mathcal{S}$ with $\mathcal{T}_2$, two agents would have to at least disagree (fundamentally let's say) on two arguments in order for this emerge.[2]

How can we make this intuition precise? We now investigate two possible ways in which the simple diff-based approach can be refined in order to take this into account. We will see that the first one, although intuitive, is flawed.

6.1 Critical subsets approach

The first idea comes from a concept introduced by Gabbay [16]. Instead of looking at all arguments, one specifically focuses on the *critical subsets*.

Definition 7. Given an AF $\mathcal{A} = (Ar, \rightharpoonup)$, a subset $X \subseteq Ar$ is *critical* iff for any $\mathcal{L}_1$, $\mathcal{L}_2 \in \mathit{Comp}_{\mathcal{A}}$ we get $\mathcal{L}_1 = \mathcal{L}_2$ whenever $\mathcal{L}_1$ and $\mathcal{L}_2$ agree on the arguments in X. We denote the set of critical subsets for $\mathcal{A}$ by $\mathit{crit}(\mathcal{A})$.

In other words a critical subset for $\mathcal{A}$ is a set of arguments whose status is enough to determine the status of *all* the arguments in Ar. Clearly at least one critical subset will always exist, for Ar is obviously critical. We are interested in the *minimal* critical subsets.

[2]A similar intuition to this can be found in [5] in the context of reasoning about action and belief update. The idea there is that there might exist some *causal* links between the value of one literal and that of another, which should be taken into account when calculating how much one possible world, i.e., conjunction of literals, differs from another. If the change in value of one literal is caused by another, then this change should not count towards calculating the difference.

Definition 8. We denote the collection of set-theoretically minimal subsets of $\mathit{crit}(\mathcal{A})$ by $\mathit{mincrit}(\mathcal{A})$, i.e.,
$\mathit{mincrit}(\mathcal{A}) \stackrel{\text{def}}{=} \{X \in \mathit{crit}(\mathcal{A}) \mid \nexists Y (Y \in \mathit{crit}(\mathcal{A}) \wedge Y \subset X)\}$.

If we look at the AF of Fig. 3 one can check that one critical subset is $X_1 = \{C, E, G\}$, since, the label of E (respectively G) determines the label of F (respectively H), while the label of C determines the labels of A, B and D. Indeed if C is `in` then A, B and D must all be `out`, if C is `out` then A, B, D must all be `in`, while if C is `undec` then A, B, D must all be `undec` too.

So, the first idea would be, given a basic label difference measure diff, to pick some minimal critical subset X and then just define, for all $\mathcal{S}, \mathcal{T} \in \mathit{Comp}_{\mathcal{A}}$, $d'(\mathcal{S},\mathcal{T}) = d_X(\mathcal{S},\mathcal{T})$, where

$$d_X(\mathcal{S},\mathcal{T}) \stackrel{\text{def}}{=} \sum_{A \in X} \mathit{diff}(\mathcal{S}(A), \mathcal{T}(A)). \tag{2}$$

Formally, the *critical sets distance method cd* is defined via a function $\mathbb{C}$ which selects for each $\mathcal{A}$ an element of $\mathit{mincrit}(\mathcal{A})$ and then sets $\mathit{cd}(\mathcal{S},\mathcal{T}) = d_{\mathbb{C}(\mathcal{A})}(\mathcal{S},\mathcal{T})$ for any $\mathcal{S}, \mathcal{T} \in \mathit{Comp}_{\mathcal{A}}$.

Example 2. Taking the complete labellings $\mathcal{S}$ and $\mathcal{T}_1, \mathcal{T}_2$ in Fig. 3, and taking $\mathbb{C}(\mathcal{A}) = \{C, E, G\}$ we get $\mathit{cd}(\mathcal{S},\mathcal{T}_1) = 2 \times \mathit{diff}(\texttt{in},\texttt{out})$ and $\mathit{cd}(\mathcal{S},\mathcal{T}_2) = \mathit{diff}(\texttt{in},\texttt{out})$. Thus $\mathcal{T}_2$ is deemed closer to $\mathcal{S}$ than $\mathcal{T}_1$ is.

The distance function d_X in (2) actually fares rather well when measured against the properties for distance functions from earlier, provided diff is sufficiently well-behaved:

THEOREM 1. *Let $X \in \mathit{crit}(\mathcal{A})$ and let d_X be defined from diff as in (2). Then d_X satisfies* **(dm1)** *and* **(dm3)**. *Furthermore:*
(i). *If diff satisfies* **(diff 3+)** *then d_X satisfies* **(dm5)** *(and hence also* **(dm2)***).*
(ii). *If diff satisfies* **(diff 3+)** *and* **(diff 5)** *then d_X satisfies* **(dm5+)**.
(iii). *If diff satisfies* **(diff 3++)** *then d_X satisfies* **(dm4)**.

PROOF. (*Outline*) **(dm1)** and **(dm3)** follow immediately from **(diff 1)** and **(diff 3)** respectively.
(i). First it is easy to check that if $\mathcal{T}_1 \leq_{\mathcal{S}} \mathcal{T}_2$ then, for all $A \in X$ (in fact for all $A \in Ar$),

$$\mathit{diff}(\mathcal{S}(A), \mathcal{T}_1(A)) \leq \mathit{diff}(\mathcal{S}(A), \mathcal{T}_2(A)) \tag{3}$$

(since either the left-hand side equals 0 or both sides are equal). If moreover $\mathcal{T}_1 <_{\mathcal{S}} \mathcal{T}_2$ then $\mathcal{T}_1 \neq \mathcal{T}_2$ and so, since X is critical, there exists $A^* \in X$ such that $\mathcal{T}_1(A^*) \neq \mathcal{T}_2(A^*)$. From this and $\mathcal{T}_1 \leq_{\mathcal{S}} \mathcal{T}_2$ we know $\mathcal{T}_1(A^*) = \mathcal{S}(A^*)$, hence $\mathit{diff}(\mathcal{S}(A^*), \mathcal{T}_1(A^*)) = 0 < \mathit{diff}(\mathcal{S}(A^*), \mathcal{T}_2(A^*))$ (the last inequality following from **(diff 3+)**). Hence the inequality (3) is strict for at least one argument in X and thus $d_X(\mathcal{S},\mathcal{T}_1) < d_X(\mathcal{S},\mathcal{T}_2)$.
(ii). If $\mathcal{T}_1 \leq^b_{\mathcal{S}} \mathcal{T}_2$ then, for all $A \in X$ (in fact all $A \in Ar$), either *(a)* $\mathcal{T}_1(A) = \mathcal{S}(A)$, or *(b)* $\mathcal{T}_1(A) = \mathcal{T}_2(A)$, or *(c)* $\mathcal{T}_1(A) = \texttt{undec}$ and [$(\mathcal{S}(A) = \texttt{in}$ and $\mathcal{T}_2(A) = \texttt{out})$ or vice versa]. In cases *(a)*, *(b)* inequality (3) holds as in part *(i)* above, while in *(c)* we get a strict inequality due to **(diff 5)**. If $\mathcal{T}_1 <^b_{\mathcal{S}} \mathcal{T}_2$ then $\mathcal{T}_1 \neq \mathcal{T}_2$ so, since X is critical there is some $A^* \in X$ such that $\mathcal{T}_1(A^*) \neq \mathcal{T}_2(A^*)$. Then either we are in the same situation as in *(i)* above, or case *(c)* obtains. Either way the inequality (3) will be strict for A^* and so $d_X(\mathcal{S},\mathcal{T}_1) < d_X(\mathcal{S},\mathcal{T}_2)$.

(iii). Follows from the fact that **(diff 3++)** ensures *diff* itself satisfies the triangle inequality, which lifts straightforwardly to d_X. □

Note the above result holds taking X to be *any* critical subset, not only the minimal ones. By taking $X = Ar$ we thus obtain Propositions 1-4 from Section 5 as corollaries.

One problem is that more than one minimal critical subset may exist. For example in the above example one can check that another minimal critical subset can be obtained by exchanging A for D to obtain $X_2 = \{A, E, G\}$. Indeed one can exchange any argument in the leftmost component. One could also replace E by F or G by H. We would like the distance (or at least the similarity ordering induced by it) to be independent of the particular minimal critical subset we use. Is it possible that we might get $d_{X_1}(\mathcal{S}, \mathcal{T}_1) \neq d_{X_2}(\mathcal{S}, \mathcal{T}_2)$ for different minimal critical subsets X_1, X_2? In the above example the answer is no, but unfortunately this does not always hold in general, as the next example shows.

Example 3. Let us return to the AF $\mathcal{A}_1$ depicted in Fig. 2. It is not the case that by knowing the label of one argument we know the full complete labelling, however, one can check that if we know the label of any *pair* of arguments, we automatically know the label of the third. Thus we have $mincrit(\mathcal{A}_1) = \big\{\{A, B\}, \{A, C\}, \{B, C\}\big\}$. We have $d_{\{A,B\}}(\mathcal{L}_1, \mathcal{L}_2) = 2 \times diff(\mathtt{in}, \mathtt{out})$ and $d_{\{A,B\}}(\mathcal{L}_1, \mathcal{L}_3) = diff(\mathtt{in}, \mathtt{out})$. Thus if we focus on the critical subset $\{A, B\}$ we obtain that $\mathcal{L}_3$ is closer to $\mathcal{L}_1$ than $\mathcal{L}_2$ is. But if instead we focus on critical subset $\{A, C\}$ we obtain the opposite conclusion, for $d_{\{A,C\}}(\mathcal{L}_1, \mathcal{L}_2) = diff(\mathtt{in}, \mathtt{out})$ and $d_{\{A,C\}}(\mathcal{L}_1, \mathcal{L}_3) = 2 \times diff(\mathtt{in}, \mathtt{out})$.

This sensitivity to the choice of critical subset is somewhat undesirable. Furthermore, as we will see next, even though *cd* can easily be made to satisfy all the distance properties we have presented thus far, there are some other, highly intuitive, postulates that it fails to validate.

6.2 Symmetry properties

The next distance properties we propose come from symmetry considerations. The idea is that applying the distance measure over AFs which are in some sense *equivalent* should yield equivalent results. In the context of argumentation semantics, such a property has been referred to as the language independence principle [3]. We are interested in describing a similar property in the context of distance measures. We begin with the common idea of graph-isomorphism, applied to argumentation frameworks.

Definition 9. Let $\mathcal{A}_1 = (Ar_1, \rightharpoonup_1)$ and $\mathcal{A}_2 = (Ar_2, \rightharpoonup_2)$ be two AFs. An *isomorphism from $\mathcal{A}_1$ to $\mathcal{A}_2$* is any bijection $g : Ar_1 \to Ar_2$ such that, for all $A, B \in Ar_1$, $A \rightharpoonup_1 B$ iff $g(A) \rightharpoonup_2 g(B)$. In the special case when $\mathcal{A}_1 = \mathcal{A}_2$ we call g an *automorphism*.

So basically an isomorphism just changes the names of arguments – or in the case of automorphism permutes them – while preserving the attack structure. Of course if g is an isomorphism from $\mathcal{A}_1$ to $\mathcal{A}_2$ then g^{-1} is an isomorphism from $\mathcal{A}_2$ to $\mathcal{A}_1$.

If g is an isomorphism from $\mathcal{A}_1$ to $\mathcal{A}_2$ then we can extend g to a function which converts any labelling $\mathcal{S}$ for $\mathcal{A}_1$ into a labelling $g(\mathcal{S})$ for $\mathcal{A}_2$. We define labelling $g(\mathcal{S})$ simply by taking $[g(\mathcal{S})](A) = \mathcal{S}(g^{-1}(A))$ for all $A \in Ar_2$.

PROPOSITION 5. *Let g be an isomorphism from $\mathcal{A}_1$ to $\mathcal{A}_2$. If $\mathcal{S} \in Comp_{\mathcal{A}_1}$ then $g(\mathcal{S}) \in Comp_{\mathcal{A}_2}$.*

The following property says that the distance should be the same for isomorphic AFs. This is in line with the intuition that an argument is characterised completely by its interactions with the other arguments.

(Iso) If g is an isomorphism from $\mathcal{A}_1$ to $\mathcal{A}_2$ then $d_{\mathcal{A}_1}(\mathcal{S}, \mathcal{T}) = d_{\mathcal{A}_2}(g(\mathcal{S}), g(\mathcal{T}))$

Note that this property differs from our previous distance properties in that whereas they dealt with a *fixed* AF $\mathcal{A}$ as given, this rule relates distance between labellings over *different*, but related, argumentation frameworks. Technically speaking, while all the previous rules are properties of the *labelling distance* $d_{\mathcal{A}}$ for fixed $\mathcal{A}$, **(Iso)** is a property of the *distance method*, i.e., the mapping $\mathcal{A} \mapsto d_{\mathcal{A}}$ (Definition 4). In the case of automorphism we get the special case:

(Auto) If g is an automorphism on $\mathcal{A}$ then $d_{\mathcal{A}}(\mathcal{S}, \mathcal{T}) = d_{\mathcal{A}}(g(\mathcal{S}), g(\mathcal{T}))$

The distance measure *cd* fails even to satisfy **(Auto)**, as the following example shows:

Example 4. Consider $\mathcal{A}_1$ in Fig. 2 and consider the mapping g such that $g(A) = B$, $g(B) = C$ and $g(C) = A$. It is easy to see that g is an automorphism on $\mathcal{A}_1$. Assume $\mathbb{C}(\mathcal{A}_1) = \{A, B\}$. Recall $\mathcal{L}_1 = \{(A, \mathtt{in}), (B, \mathtt{out}), (C, \mathtt{out})\}$ and $\mathcal{L}_3 = \{(A, \mathtt{out}), (B, \mathtt{out}), (C, \mathtt{in})\}$. So $g(\mathcal{L}_1) = \{(A, \mathtt{out}), (B, \mathtt{in}), (C, \mathtt{out})\}$ and $g(\mathcal{L}_3) = \{(A, \mathtt{in}), (B, \mathtt{out}), (C, \mathtt{out})\}$. Then if **(Auto)** were satisfied we would expect $cd(\mathcal{L}_1, \mathcal{L}_3) = d_{\{A,B\}}(\mathcal{L}_1, \mathcal{L}_3) = d_{\{A,B\}}(g(\mathcal{L}_1), g(\mathcal{L}_3))$, but $d_{\{A,B\}}(\mathcal{L}_1, \mathcal{L}_3) = diff(\mathtt{in}, \mathtt{out}) \neq 2 \times diff(\mathtt{in}, \mathtt{out}) = d_{\{A,B\}}(g(\mathcal{L}_1), g(\mathcal{L}_3))$. Note this example assumes $\mathbb{C}(\mathcal{A}_1) = \{A, B\}$, but it should be clear that counterexamples can also be found if either of the other two elements of $mincrit(\mathcal{A}_1)$ were selected.

Summarising this section so far, we have managed to find a distance method *cd* which respects the intuitions of the example of Fig. 3, but at the expense of violating what seem to be a highly desirable postulates (**(Iso)** and **(Auto)**) for distance methods. Is there a distance method which can satisfy *all* our desiderata? We shall now see that the answer is yes.

6.3 Distance via issue-wise label difference

We want to capture the idea that the labels of two arguments are "tied together". For example in a simple 2-argument AF consisting of two arguments A and B mutually attacking each other, there may be two arguments but to all intents and purposes there is really only one "issue" at stake, and that is whether A or B (or neither) should be accepted. We want to isolate these different issues which are being argued over. Given an AF $\mathcal{A} = (Ar, \rightharpoonup)$, let us define the following two binary relations over Ar. For any $A, B \in Ar$:

- $A \equiv_1 B$ iff $\forall \mathcal{L} \in Comp_{\mathcal{A}} : \mathcal{L}(A) = \mathcal{L}(B)$
- $A \equiv_2 B$ iff $\forall \mathcal{L} \in Comp_{\mathcal{A}} : (\mathcal{L}(A) = \mathtt{in} \Leftrightarrow \mathcal{L}(B) = \mathtt{out}) \wedge (\mathcal{L}(A) = \mathtt{out} \Leftrightarrow \mathcal{L}(B) = \mathtt{in})$.

$A \equiv_1 B$ means that the labels assigned to A and B are exactly the same in all complete labellings, i.e., that A and B are in a sense logically equivalent, while $A \equiv_2 B$ means

that A and B always receive "opposite" labels: whenever A is labelled `in` then B is labelled `out`, and vice versa. It is easy to see that if $A \equiv_2 B$ then we also have $\mathcal{L}(A) = \texttt{undec}$ iff $\mathcal{L}(B) = \texttt{undec}$. From these two relations we define

$$A \equiv B \text{ iff } (A \equiv_1 B \vee A \equiv_2 B).$$

Thus if $A \equiv B$ then intuitively the labels of A and B are "in sync", in that the label of one cannot be changed without causing a change of equal magnitude to the label of the other.

PROPOSITION 6. *$\equiv$ is an equivalence relation over Ar.*

PROOF. *(Outline).* Reflexivity holds since $\equiv_1$ is reflexive. Symmetry holds since both $\equiv_1$ and $\equiv_2$ are symmetric, and transitivity holds because of the following composition properties: $(\equiv_1 \circ \equiv_1) = (\equiv_2 \circ \equiv_2) = \equiv_1$ and $(\equiv_1 \circ \equiv_2) = (\equiv_2 \circ \equiv_1) = \equiv_2$. □

Within each $\equiv$-equivalence class, there are at most 3 possible labellings which can occur: either *(i)* all its elements are labelled `undec`, or *(ii)* all its elements are set to `in` or `out`, or *(iii)* the "inverse" labelling to *(ii)* occurs, in which those arguments labelled `in` become `out` and those labelled `out` are now `in`. Essentially each equivalence class acts as a single 3-valued argument. We call each such class an *issue* of the given AF.

Definition 10. Given an AF $\mathcal{A} = (Ar, \rightharpoonup)$, the set $\mathbb{I}(\mathcal{A})$ of *issues* of $\mathcal{A}$ is defined as $\mathbb{I}(\mathcal{A}) = Ar/\equiv$. For $A \in Ar$ we will denote the $\equiv$-equivalence class of A by $[A]$.

For example, it can be checked that the issues for the AF in Fig. 3 are $\{A, B, C, D\}$, $\{E, F\}$ and $\{G, H\}$. In the AF of Fig. 2, however, there are 3 issues $\{A\}$, $\{B\}$ and $\{C\}$. Note that in the former case the issues coincide exactly with the *strongly connected components* [4] of the graph, whereas this is not true of the latter case.

Now, rather than calculate distance via argument-wise label difference as we did in Section 5, we can instead do it via *issue-wise* label difference. For this we need to define the measure of disagreement $DIFF(\mathcal{S}, \mathcal{T}, [A])$ between two labellings $\mathcal{S}$ and $\mathcal{T}$ on a single issue $[A]$. We do this using a basic label difference measure *diff*:

$$DIFF(\mathcal{S}, \mathcal{T}, [A]) \stackrel{\text{def}}{=} diff(\mathcal{S}(A), \mathcal{T}(A)).$$

PROPOSITION 7. *DIFF is well-defined, i.e., if $[A] = [B]$ then $diff(\mathcal{S}(A), \mathcal{T}(A)) = diff(\mathcal{S}(B), \mathcal{T}(B))$.*

PROOF. *(Outline)* If $[A] = [B]$ then either $A \equiv_1 B$ or $A \equiv_2 B$. In the former case the result is clear. In the latter case one may simply check for each of the 9 possibile combinations of labels for $\mathcal{S}(A)$, $\mathcal{T}(A)$. E.g., if $\mathcal{S}(A) = \texttt{in}$ and $\mathcal{T}(A) = \texttt{undec}$ then, from $A \equiv_2 B$ we know $\mathcal{S}(B) = \texttt{out}$ and $\mathcal{T}(B) = \texttt{undec}$. Thus $diff(\mathcal{S}(A), \mathcal{T}(A)) = diff(\mathcal{S}(B), \mathcal{T}(B))$ since *diff* by **(diff 4)**. □

Note that this result depends on the assumptions that *diff* satisfies **(diff 1)**, **(diff 2)** and **(diff 4)**.

Finally the issue-based distance measure *id* is defined by setting, for any $\mathcal{S}, \mathcal{T} \in Comp_{\mathcal{A}}$,

$$id(\mathcal{S}, \mathcal{T}) = \sum_{[A] \in \mathbb{I}(\mathcal{A})} DIFF(\mathcal{S}, \mathcal{T}, [A])$$

In the example in Fig. 3 we have $id(\mathcal{S}, \mathcal{T}_1) = 2 \times diff(\texttt{in}, \texttt{out})$ and $id(\mathcal{S}, \mathcal{T}_2) = diff(\texttt{in}, \texttt{out})$, as with the critical subsets approach of Section 6.1. For the example in Fig. 2 we get $id(\mathcal{L}_1, \mathcal{L}_2) = 2 \times diff(\texttt{in}, \texttt{out}) = id(\mathcal{L}_1, \mathcal{L}_3)$. So according to the issue-based distance $\mathcal{L}_2$ and $\mathcal{L}_3$ are equidistant from $\mathcal{L}_1$.

The issue-wise distance measure can be related to the preceding critical subsets approach. Clearly we have $id(\mathcal{S}, \mathcal{T}) = d_X(\mathcal{S}, \mathcal{T})$ (see equation (2) in Section 6.1), where X is any set formed by taking a representative of each $\equiv$-equivalence class. It turns out that we have the following:

PROPOSITION 8. *Let X be any set obtained by taking one element of each issue in $\mathbb{I}(\mathcal{A})$. Then $X \in crit(\mathcal{A})$.*

PROOF. *(Outline)* Let $\mathcal{L}_1$, $\mathcal{L}_2 \in Comp_{\mathcal{A}}$ be 2 complete labellings which agree on X. We must show $\mathcal{L}_1(A) = \mathcal{L}_2(A)$ for all $A \in Ar$. Let $A^* \in X$ be the chosen representative of $[A]$ in X, so $A^* \equiv A$. We know $\mathcal{L}_1(A^*) = \mathcal{L}_2(A^*)$. Denote this common label by `x`. If $A^* \equiv_1 A$ then both $\mathcal{L}_1(A)$ and $\mathcal{L}_2(A)$ are equal to `x` as required. Suppose $A^* \equiv_2 A$. If $\texttt{x} = \texttt{in}$ then both $\mathcal{L}_1(A)$ and $\mathcal{L}_2(A)$ are `out`. If $\texttt{x} = \texttt{out}$ then both $\mathcal{L}_1(A)$ and $\mathcal{L}_2(A)$ are `in`. Finally if $\texttt{x} = \texttt{undec}$ then both $\mathcal{L}_1(A)$ and $\mathcal{L}_2(A)$ are `undec`. □

Thus *id* can be thought of as a critical-set based distance which chooses from among a particular class of critical sets, viz. those which contain one argument from each issue. Furthermore, unlike the critical-set based distance the precise choice of these elements is irrelevant. However the critical set chosen need not be a minimal one, i.e., an element of $mincrit(\mathcal{A})$, as can be seen already in the AF of Fig. 2. We may deduce from all this and Theorem 1 the following:

THEOREM 2. *id satisfies* **(dm1)** *and* **(dm3)**. *Furthermore:*
(i). *If diff satisfies* **(diff 3+)** *then id satisfies* **(dm5)** *(and hence also* **(dm2)***).*
(ii). *If diff satisfies* **(diff 3+)** *and* **(diff 5)** *then id satisfies* **(dm5+)**.
(iii). *If diff satisfies* **(diff 3++)** *then id satisfies* **(dm4)**.

In addition, we have the following:

THEOREM 3. *The distance method $\mathcal{A} \mapsto id_{\mathcal{A}}$ satisfies* **(Iso)** *(and hence also* **(Auto)***).*

PROOF. *(Outline.)* Let $\mathcal{A}_1, \mathcal{A}_2$ be 2 AFs connected by isomorphism g and let $\mathcal{S}, \mathcal{T} \in Comp_{\mathcal{A}_1}$. To show $id_{\mathcal{A}_1}(\mathcal{S}, \mathcal{T}) = id_{\mathcal{A}_2}(g(\mathcal{S}), g(\mathcal{T}))$ we show that the summands on each side of this identity match up in pairs. More precisley we show there is a bijection $h : \mathbb{I}(\mathcal{A}_1) \to \mathbb{I}(\mathcal{A}_2)$ such that, for each $[A] \in \mathbb{I}(\mathcal{A}_1)$, $DIFF(\mathcal{S}, \mathcal{T}, [A]) = DIFF(g(\mathcal{S}), g(\mathcal{T}), h([A]))$. Indeed we can just define $h([A]) = [g(A)]$. The facts that h is well-defined and injective are both proved using the property that, for any $A, B \in Ar_1$, A and B belong to the same issue in $\mathbb{I}(\mathcal{A}_1)$ iff $g(A)$ and $g(B)$ belong to the same issue in $\mathbb{I}(\mathcal{A}_2)$. h is clearly surjective since, given any $[Z] \in \mathbb{I}(\mathcal{A}_2)$ we have $[Z] = h([g^{-1}(Z)])$. Finally $DIFF(g(\mathcal{S}), g(\mathcal{T}), h([A])) = DIFF(g(\mathcal{S}), g(\mathcal{T}), [g(A)]) = diff([g(\mathcal{S})](g(A)), [g(\mathcal{T})](g(A)))$. Since $[g(\mathcal{S})](g(A)) = \mathcal{S}(g^{-1}(g(A))) = \mathcal{S}(A)$ by definition of $g(\mathcal{S})$ (and similarly for $\mathcal{T}$), this equals $diff(\mathcal{S}(A), \mathcal{T}(A)) = DIFF(\mathcal{S}, \mathcal{T}, [A])$ as required. □

7. RELATED WORK AND CONCLUSION

We have initiated the investigation of the notion of distance between two reasonable evaluations of an argument graph. While this issue has been investigated in non-argument based accounts of both belief revision [18, 20], in judgement

aggregation [19, 21], and in abstract preferences [1], to our knowledge we are the first to study it in the context of formal argumentation theory.

We presented several different distance functions, all defined on top of a difference function on the space of possible labels $\{\texttt{in}, \texttt{out}, \texttt{undec}\}$. These functions fall into two groups: those which sum the difference between the labels of *all* arguments Ar in the framework, and those which single out various subsets of Ar as being in some sense the *critical* ones. We gave some postulates for such distance functions, even though we saw that many simple and straightforward candidates for distance measures suffer from some problem or another, and we developed some intuitions via several examples about what a distance function between complete labellings should be like.

For future work we would like to investigate more closely the issue-based distance method *id*. Specifically we are looking for other properties that it satisfies, perhaps leading to an *axiomatic characterisation*. We also want to apply these new distances to the problems of revision and judgement aggregation in argumentation. In revision we want to choose the closest labelling to the current one which extends an input new partial labelling. In questions of judgement aggregation we want to choose the labellings which are closest to the group as a whole. Similar considerations have been applied in propositional contexts (e.g. [13, 17]), while a first exploration of the use of Hamming-like distances (see Section 5) in labelling-aggregation has been carried out by Caminada et al. in [12], where it is used to check the manipulability and Pareto optimality of certain aggregation operators. They assume each member of a group of agents provides a complete labelling, and that each agent's preference relation over the set of all complete labellings is given by Hamming set or Hamming distance from its given labelling.

It would also be interesting to see if the issue-based methodology of Section 6.3 can be used to refine the distance-based approaches already existing in general judgement aggregation. Finally, here we focused on complete labellings. This is reasonable since they correspond to rational, coherent standpoints. But the definitions will work for other families of labellings too, like preferred, stable [14], and semi-stable [9].

8. ACKNOWLEDGEMENTS

Thanks are due to the reviewers for their encouraging remarks. Thanks also to Ringo Baumann, Gerhard Brewka and the Individual and Collective Reasoning group at the University of Luxembourg for some useful comments. Richard Booth is supported by the FNR/INTER project "Dynamics of Argumentation". Martin Caminada and Mikołaj Podlaszewski are supported by the National Research Fund, Luxembourg (FNR) (LAAMI and LAAMIcomp projects).

9. REFERENCES

[1] N. Baigent. Preference proximity and anonymous social choice. *The Quarterly Journal of Economics*, 102(1):161–169, 1987.

[2] P. Baroni, M. Caminada, and M. Giacomin. An introduction to argumentation semantics. *Knowledge Engineering Review*, 26:365–410, 2011.

[3] P. Baroni and M. Giacomin. On principle-based evaluation of extension-based argumentation semantics. *Artificial Intelligence*, 171(10-15):675–700, 2007.

[4] P. Baroni, M. Giacomin, and G. Guida. SCC-recursiveness: A general schema for argumentation semantics. *Artificial Intelligence*, 168(1-2):162–210, 2005.

[5] G. Brewka and J. Hertzberg. How to do things with worlds: On formalizing actions and plans. *Journal of Logic and Computation*, 3(5):517–532, 1993.

[6] M. Caminada. On the issue of reinstatement in argumentation. In *Proc. JELIA*, pages 111–123, 2006.

[7] M. Caminada. Comparing two unique extension semantics for formal argumentation: ideal and eager. In *Proc. BNAIC*, pages 81–87, 2007.

[8] M. Caminada and L. Amgoud. On the evaluation of argumentation formalisms. *Artificial Intelligence*, 171(5-6):286–310, 2007.

[9] M. Caminada, W. Carnielli, and P. Dunne. Semi-stable semantics. *Journal of Logic and Computation*, 2011. in print.

[10] M. Caminada and D. Gabbay. A logical account of formal argumentation. *Studia Logica*, 93(2):109–145, 2009.

[11] M. Caminada and G. Pigozzi. On judgment aggregation in abstract argumentation. *Autonomous Agents and Multi-Agent Systems*, 22(1):64–102, 2011.

[12] M. Caminada, G. Pigozzi, and M. Podlaszewski. Manipulation in group argument evaluation. In *Proc. IJCAI*, pages 121–126, 2011.

[13] M. Dalal. Investigations into a theory of knowledge base revision. In *Proc. AAAI*, pages 475–479, 1988.

[14] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[15] P. M. Dung, P. Mancarella, and F. Toni. Computing ideal sceptical argumentation. *Artificial Intelligence*, 171(10-15):642–674, 2007.

[16] D. Gabbay. Fibring argumentation frames. *Studia Logica*, 93(2):231–295, 2009.

[17] S. Konieczny, J. Lang, and P. Marquis. DA^2 merging operators. *Artificial Intelligence*, 157(1-2):49–79, 2004.

[18] D. Lehmann, M. Magidor, and K. Schlechta. Distance semantics for belief revision. *Journal of Symbolic Logic*, 66(1):295–317, 2001.

[19] M. Miller and D. Osherson. Methods for distance-based judgment aggregation. *Social Choice and Welfare*, 32(4):575–601, 2009.

[20] P. Peppas, S. Chopra, and N. Foo. Distance semantics for relevance-sensitive belief revision. In *Proc. KR*, pages 319–328, 2004.

[21] G. Pigozzi. Belief merging and the discursive dilemma: an argument-based account to paradoxes of judgment aggregation. *Synthese*, 152(2):285–298, 2006.

[22] H. Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1(2):93–124, 2010.

[23] I. Rahwan and F. Tohmé. Collective argument evaluation as judgement aggregation. In *Proc. AAMAS*, pages 417–424, 2010.

[24] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

Cooperative Dialogues with Conditional Arguments

Samy Sá
Universidade Federal do Ceará
Campus do Pici, Bl 910
Fortaleza, Brazil
samy@ufc.br

João Alcântara
Universidade Federal do Ceará
Campus do Pici, Bl 910
Fortaleza, Brazil
jnando@lia.ufc.br

ABSTRACT

We introduce an approach to cooperative dialogues as a framework for group deliberation. One of its distinguishing features is that it deals with conditional and constraint-based arguments, which are built by employing abductive and hypothetical reasoning. These kinds of arguments allow agents to use a variety of dialogue moves proper to a cooperative debate, such as argument rewrites and conditional attacks. In our approach, a group of agents develops a dialogue as they explore different lines of thought to build a group position in a yes or no decision. In essence, given a matter for discussion, the parties involved will consider arguments that either supports or rejects it and discuss such arguments to decide whether or not to accept them. To achieve that, agents will work as a team and combine their knowledge to produce more complex arguments and study possible flaws these might have.

Categories and Subject Descriptors

F.4.1 [**Mathematical Logic**]: Logic and constraint programming; I.2.11 [**Distributed Artificial Intelligence**]: Multiagent systems

General Terms

Theory

Keywords

Collective Intelligence, Reasoning (single and multiagent), Argumentation, Logic Programming, Abduction

1. INTRODUCTION

Dialogues were introduced into multiagent systems to formalize the generation and interpretation of arguments exchanged amongst agents [1]. Such dialogues are perceived as a game involving two antagonistic agents in a discussion about some matter (a proposition). In this setting, the first player tries to justify or defend the matter, while the second will try to disqualify or attack it. However, we consider that some dialogues are inherently cooperative in the sense that the agents in a group might share a goal and be interested in working together to justify or disqualify a proposition. This is the case with Deliberation Dialogues [14], which is our main focus. We believe that the key for this kind of cooperation lies in abductive reasoning [7], which is a special kind of non-monotonic reasoning, usually defined as inference to the best explanation.

In this paper we focus on deliberation dialogues about whether the group can explain a scenario or if they should accept an argument from an external source. Our goal is to allow agents to collectively engage in reasoning about what to do, however without the need to share their entire knowledge bases. This is an important feature, since in human-agent interaction, knowledge bases cannot be simply merged. We can also think of a self-preservation rationality, for an agent might experience disadvantages if it later gets into a negotiation or game involving an agent with whom it just shared its entire knowledge base. On the other hand, the kind of deliberation we propose is hardly as efficient as joining the knowledge bases to draw collective conclusions.

In existing approaches to dialogues in multiagent systems [1, 14], the agents in a group will share opinions (arguments) to reach a consensus on which ones are good. Their knowledge, however, is combined in a very limited way because every opinion is proposed by a single agent. In our work, agents engage in hypothetical reasoning to consider alternative scenarios and combine their knowledge further. As a consequence, agents can cooperate to complement the arguments from one another and reach a deeper understanding of the possible flaws their arguments might have.

Our work defines a framework for the exchange of arguments between agents, such as in multiagent dialogues [1] and negotiation with abduction [17]. Multiagent dialogues are characterized in [1] as a game involving two agents in antagonistic positions about some matter of discussion, possibly a deal in a process of negotiation. Agents will place arguments attacking each other opinions until one of them can no longer respond, so the other will be the winner. Similarly, negotiation is perceived as an exchange of proposals, and whenever an agent can no longer counter the last, it has to accept it. Negotiation was improved with abduction in [17], where the authors introduce the possibility of conditional proposals based on abductive reasoning. In cooperative deliberation as we introduce in this paper, agents work as team mates and explore alternatives by exchange arguments for the best interest of the group.

Amongst others, our framework has the following characteristics: (i) agents can resort to hypothetical reasoning to produce arguments; (ii) the agents in a group might be

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

able to combine their knowledge to produce elaborate arguments that no single agent can conceive on its own; (*iii*) the reasoning performed by a group of agents involves two opposing consistent positions; (*iv*) the dialogues are guaranteed to end, so the agents are sure to reach an agreement about the subject of discussion.

In Section 2, we will present abductive logic programs as they are used throughout the paper. Next, we will introduce conditional arguments in Section 3 and our approach to collective dialogues in Section 4. These last two sections hold our main contributions. We discuss related work in Section 5 and conclude the paper in Section 6 with a discussion on the importance of our contributions and future work.

2. PRELIMINARIES

2.1 Extended Disjunctive Programs

In this paper, we account for programs as in Extended Disjunctive Programs (EDP's) [10] without disjunctive heads.

An EDP is defined over a *Herbrand Universe* HB, the set of all ground atoms the program might resort to. Such a program consists of a set of rules of the form

$$r: \ L_H \leftarrow L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$$

with L_H being optional and $n \geq m \geq 0$. In this notation, each L_i is a literal (an atom A or its negation $\neg A$), L_H is a literal, and *not* is *negation as failure* (NAF). If L is a literal, *not* L is called a NAF-*Literal*. We might speak of literals to generalize literals and NAF-Literals. In a rule r on the above form, we refer to L_H as the *head* of the rule and write $head(r)$ to denote the set $\{L_H\}$. We refer to the conjunction $L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$ as the body of r, and $body(r)$ denotes the set $\{L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n\}$. We differ the literals of its positive and negative parts as $body^+(r)$ and $body^-(r)$ to refer to the sets $\{L_1, \ldots, L_m\}$ and $\{L_{m+1}, \ldots, L_n\}$, respectively. We also denote $not\_body^-(r)$ as the set of NAF-*Literals* $\{not\ L_{m+1}, \ldots, not\ L_n\}$. A rule may be written as $head(r) \leftarrow body^+(r), not\_body^-(r)$ or $head(r) \leftarrow body(r)$, for $body(r) = body^+(r) \cup not\_body^-(r)$. A rule is an integrity constraint if $head(r) = \emptyset$ and it is a fact if $body(r) = \emptyset$, in which case we do not write "$\leftarrow$". We say a program is NAF-free if it does not contain NAF-Literals.

The semantics of an EDP is given by the Answer Sets Semantics [10]. Consider Lit_P is the set of all literals in the language of a program P and S one of its subsets. Let P^S be the set that contains all the instances $head(r) \leftarrow body^+(r)$ of rules of P such that $body^-(r) \cap S = \emptyset$ and no other rules, so P^S is a NAF-free program. Given a NAF-*free* EDP P, $Ans(P)$ is a minimal subset of Lit_P such that (*i*) for every ground rule of P, if $body^+(r) \subseteq S$, then $head(r) \in S$ and (*ii*) S is either consistent or $S = Lit_P$. Given an EDP P, S will be an answer set of P if $S = Ans(P^S)$. A program might have zero, one or multiple answer sets. An answer set S for P is consistent if S does not simultaneously contain A and $\neg A$, for no atom in the language. The program itself will be said consistent if it has a consistent answer set. Otherwise, the program is inconsistent.

We draw special attention to the following terminology:

- A *goal* is a conjunction of literals and NAF-literals. If G is a goal, then $Lit(G)$ is the set of literals and NAF-literals in G. If Hyp is a set of rules (a program), $Lit(Hyp) = \{L \in (body(r) \cup head(r)) |\ r \in Hyp\}$.
- An EDP P satisfies $G = L_1, \ldots, L_m, not\ L_{m+1}, \ldots, not\ L_n$ (written $P \models G$) if P has an answer set S such that $\{L_1, \ldots, L_m\} \subseteq S$ and $\{L_{m+1}, \ldots, L_n\} \cap S = \emptyset$.
- Given a literal or NAF-literal L, we have: if $L = A$, then $neg(L) = \neg A$; if $L = \neg A$, then $neg(L) = A$; if $L = not\ L'$, $L' \in \{A, \neg A\}$, then $neg(L) = L'$.

2.2 Abductive Logic Programs

Abduction is a special kind of non-deductive reasoning in which hypotheses are inferred to explain observable facts otherwise not accepted by a theory. Abductive Logic Programming brings this feature to standard logic programming [12, 7]. We will now introduce Abductive Logic Programs (ALP's) as in the abductive framework of *Extended Abduction* [16, 17], but adapted to our objectives.

An abductive program is a pair $\langle P, H \rangle$, where P is an EDP and H is a set of literals referred to as abducibles. If P is consistent, then $\langle P, H \rangle$ is consistent. Throughout the paper we will assume only consistent programs. A goal[1] is satisfied by $\langle P, H \rangle$ if $\{L_1, \ldots, L_m\} \subseteq S$ and $\{L_{m+1}, \ldots, L_n\} \cap S = \emptyset$ for some answer set S of P.

Definition 1. *Let G be a goal for the ALP $\langle P, H \rangle$. A pair (E, F) is an* explanation *to G in $\langle P, H \rangle$ if*

1. *$(P \setminus F) \cup E$ has an answer set which satisfies G[2],*
2. *$(P \setminus F) \cup E$ is consistent,*
3. *E and F are sets of literals such that $E \subseteq H \setminus P$ and $F \subseteq H \cap P$.*

Intuitively, an explanation (E, F) of G in $\langle P, H \rangle$ means that by assuming the literals in E as true while retracting (falsifying) the literals in F from P, the resulting $P' = (P \setminus F) \cup E$ satisfies G. If the original program has an answer set satisfying G, then $(\emptyset, \emptyset)$ is an explanation and no changes are needed in P. An explanation (E, F) is minimal if, for any explanation (E', F') such that $E' \subseteq E$ and $F' \subseteq F$, then $E' = E$ and $F' = F$. In general, only the minimal explanations are of interest.

If an agent has a program $\langle P, H \rangle$ as its knowledge base, the set H lists the literals the agent can resort to for hypothetical reasoning. These literals might not be enough to explain some goals, so an agent should be able to adapt and consider unknown literals from such goals as abducibles too. This capability might not be helpful when the agent is on its own, but it can be essential in a group deliberation setting.

Definition 2. *Given a goal G and a program $\langle P, H \rangle$, $Ab(P, H, G)$ is the set of literals that appear in G but appear neither in P nor in H. An agent adapts to deliberate on G if it considers the literals in $Ab(P, H, G)$ as abducibles, i.e., if it reasons as if its knowledge base were $\langle P, H \cup Ab(P, H, G) \rangle$.*

Example 1. *Consider the following ALPs:*

$P_1:$	$a \leftarrow b;$ $b \leftarrow c;$	$P_2:$	$b \leftarrow d;$ $d \leftarrow not\ e;$ e	$P_3:$	$\leftarrow c, d;$ $d;$
$H_1:$	$\{b, c\}$	$H_2:$	$\{d, e\}$	$H_3:$	$\{d, c\}$

[1] In [17], goal is referred as observation.

[2] This definition is for *credulous* explanations. Its choice over *skeptical* explanations [11] makes possible to have more explanations and gives us a better chance of finding good related arguments in a discussion.

Consider a group of three agents Ag_1, Ag_2, Ag_3. Each agent Ag_i has its knowledge base in ALP $\langle P_i, H_i\rangle$, $i = 1, 2, 3$. These agents can build, amongst others, the following minimal explanations (as in Definition 1):

$Ag_1: \ Ex_1 = (E_1, F_1) = (\{b\}, \{\ \})$ *for* $G_1 = a, b.$
$Ag_1: \ Ex_2 = (E_2, F_2) = (\{b, d\}, \{\ \})$ *for* $G_2 = a, d,$ *not* $c.$
$Ag_2: \ Ex_3 = (E_3, F_3) = (\{d\}, \{\ \})$ *for* $G_3 = b.$
$Ag_2: \ Ex_4 = (E_4, F_4) = (\{\ \}, \{e\})$ *for* $G_4 = d.$
$Ag_3: \ Ex_5 = (E_5, F_5) = (\{c\}, \{d\})$ *for* $G_5 = c.$

The first explanation suggests that if b is true, Ag_1 can prove $G_1 = a, b$. The second explanation uses the literal $d \in Ab(P_1, H_1, G_2)$, which is not in P_1, so the agent has adapted to deliberate on G_2 (Definition 2) and has built the explanation Ex_2 in $\langle P_1, H_1 \cup Ab(P_1, H_1, G_2)\rangle$. It means that Ag_1 cannot justify d in G_2 and that if b is true, Ag_1 can prove the part of G_2 that does not involve d, i.e., $G_2' = a,$ not c. The explanation (E_3, F_3) has a similar meaning to that of (E_1, F_1), and is highlighted because it can be combined with the latter to create a new explanation to G_1, as we will later explore in the paper. The next explanation states that Ag_2 believes e is true ($\langle P_2, H_2\rangle \models e$), but is capable to conceive e being false, as it would explain $G_4 = d$. Finally, Ag_3 could prove $G_5 = c$ under the conditions that c is true and d is false, but it believes d is true and has no opinion about c. In each case, an explanation is intended to mean that an agent is willing to discuss some of its knowledge.

Further, the goal $G_6 = a, b, d$ cannot be satisfied by any of the three programs, even though the program $\langle P_\cup, H_\cup\rangle$, where $P_\cup = \bigcup P_i$ and $H_\cup = \bigcup H_i$, $i = 1, 2, 3$, satisfies it. We highlight that if the agents adapt (Definition 2), they can produce explanations to G_6. In the next section, we will show explanations play a key role in the definition of conditional arguments. Also, we will show agents capable of abductive reasoning as above can satisfy G_6 in the example as they share and complement each others explanations.

3. CONDITIONAL ARGUMENTS

Intuitively, an argument consists of a conclusion and some justification to it. The reading of an argument is that if its justification is acceptable, the conclusion should be as well. These arguments might be somehow defective, so other arguments can be proposed to point its possible flaws. In this section, we will formalize arguments and proceed to extend this notion with hypothetical reasoning. To that sense, we will introduce two kinds of *conditional* arguments: The first kind is based on hypothetical scenarios in which a particular conclusion makes sense. Using this type of argument can enrich the discussion of a matter, as it allows agents to go deeper on exploring the possible flaws the arguments might have and to cooperate with other agents by combining their knowledge. The second kind of argument involves arguing that some hypotheses from an argument can lead to absurd conclusions, so it should be rejected.

Definition 3 (Arguments). *An argument in an EDP P is a pair (Hyp, G) where G is a goal and Hyp is a set of instances of rules of P such that (i) there is a consistent answer set of P that satisfies Hyp; (ii) $Hyp \models G$ and (iii) Hyp is minimal, so no $Hyp' \subset Hyp$ satisfies both i and ii.*

If (Hyp, G) is an argument, the set Hyp is called the support or hypotheses set and G is the *conclusion* of the argument. Given an argument $Arg = (Hyp, G)$, its set of literals and NAF-literals is $Lit(Arg) = Lit(Hyp) \cup Lit(G)$.

In a dialogue, an agent can disagree with others by attacking their arguments:

Definition 4. *An argument $Arg_1 = (Hyp_1, G_1)$ attacks $Arg_2 = (Hyp_2, G_2)$ if there is a $L_1 \in Lit(G_1)$ such that $L_1 = neg(L_2)$ for some literal $L_2 \in Lit(Arg_2)$.*

Example 2. *Consider an agent Ag with knowledge base represented by the following EDP P:*

$$P: \quad a \leftarrow not\ b;$$
$$\quad\ \ c$$

The goal $G_1 = a, c$ is satisfied by P, so Ag can produce an argument $Arg_1 = (\{a \leftarrow not\ b; c\}, G_1)$ to explain it to the other agents. The goal $G_2 = a, \neg c$, however, is not satisfied by P, and Ag can produce the argument $Arg_2 = (\{c\}, c)$ to suggest it cannot be satisfied by the group, since c denies $\neg c$ in G_2. The conclusion of $Arg_3 = (\{\neg c \leftarrow a; a\}, \neg c)$ denies an hypothesis of Arg_2, so Arg_3 attacks Arg_2.

3.1 Abduction-Based Conditional Arguments

An agent capable of abductive reasoning can conceive alternative hypothetical scenarios in which a goal would be satisfied. The agent can then build arguments in any alternative scenario and highlight the conditions in which the scenario would be acceptable. We refer to these arguments as *conditional arguments*, for they can only be accepted by a group of agents if the conditions presented in the argument are satisfied by them. A notion of conditional arguments has been introduced in [13], but the following definitions are original and based on extended abduction (Section 2.2).

Definition 5 (Conditional Arguments). *Consider $\langle P, H\rangle$, an ALP, and (E, F), a minimal explanation to the goal G. The tuple (E, F, Hyp, G) is a* conditional *argument to G if (Hyp, G) is an argument in $P' = (P \backslash F) \cup E$. An argument (E, F, Hyp, G) with $E = F = \emptyset$ is* non-conditional.

If (E, F, Hyp, G) is a conditional argument, Hyp is its support or hypothesis set, G is the *conclusion* and each element of Hyp is a hypothesis. To denote the set of conditions that the explanation adds to the argument, we write $C(E, F) = E \cup \{not\ L \mid L \in F\}$. If $Arg = (E, F, Hyp, G)$ is a conditional argument, we denote its set of literals and NAF-Literals as $Lit(Arg) = C(E, F) \cup Lit(Hyp) \cup Lit(G)$.

The idea is that a conditional argument proposed by an agent would be accepted in our framework if the explanation in it is justified by the other agents.

Example 3. *Consider an agent Ag with knowledge base represented by the following ALP $\langle P, H\rangle$:*

$$P: \quad a \leftarrow b;$$
$$\quad\ \ c;$$
$$H: \quad \{b, c\}$$

The goal $G = a, c$ is not satisfied by P, but Ag can produce the explanation $(E, F) = (\{b\}, \{\})$ and build the conditional argument $A = (\{b\}, \{\ \}, \{b;\ a \leftarrow b;\ c\},\ G)$ in $P' = P \cup \{b\}$.

The definition of attack with conditional arguments is about the same as before, though now it is also possible to attack the explanation attached to an argument.

DEFINITION 6. *An argument* $Arg_1 = (E_1, F_1, Hyp_1, G_1)$ *attacks* $Arg_2 = (E_2, F_2, Hyp_2, G_2)$ *if there is a* $L_1 \in Lit(G_1)$ *such that* $L_1 = neg(L_2)$ *for some literal* $L_2 \in Lit(Arg_2)$. *If* (E_1, F_1, Hyp_1, G_1) *is an attack to some argument and* $C(E_1, F_1) \neq \emptyset$, *we say it is a* conditional attack.

EXAMPLE 4. *Consider* $Arg_1 = (\{b\}, \{\ \}, \{b;\ a \leftarrow b;\ c\},\ G)$, $G = a, c$. *The arguments* $Arg_2 = (\{\ \}, \{\ \}, \{\neg b \leftarrow not\ d;\}, \neg b)$ *and* $Arg_3 = (\{c\}, \{\ \}, \{c;\ \neg b \leftarrow c;\}, \neg b)$ *are examples of attacks to* Arg_1 *that focus on its conditions. On top of that,* Arg_3 *is a conditional attack to* Arg_1.

3.2 Building Arguments Together

A conditional argument should only be accepted by a group of agents if its conditions are satisfied by other agents. The agents in a group should therefore cooperate to reduce the number of conditions of an argument and transform it into a non-conditional argument (Definition 5). This is done by *rewriting* a conditional argument, i.e., adding support to some of its conditions, even though it might be necessary to introduce others.

DEFINITION 7 (ARGUMENT REWRITE). *Consider a conditional argument* $Arg_1 = (E_1, F_1, Hyp_1, G_1)$ *proposed by agent* Ag_1. *Also, consider an agent* Ag_2 *can produce an argument* $Arg_2 = (E_2, F_2, Hyp_2, G_2)$ *such that* $G_2 = L_1$, *for some* $L_1 \in E_1$. *Then,* Ag_2 *rewrites* Arg_1 *as* $Arg_1{}' = ((E_1 \setminus \{L_1\}) \cup E_2,\ F_1 \cup F_2,\ (Hyp_1 \setminus \{L_1\}) \cup Hyp_2,\ G_1)$.

In our framework, an argument rewrite will consist of a dialogue move in which an agent attempts to unify its opinion with those of other agents. In particular, if $(E_2 = \emptyset$, the rewriting consists in reducing the cardinality of E_1, being therefore an attempt to fulfill the conditions of Arg_1.

EXAMPLE 5. *Consider* $Arg_1 = (\{a\}, \{\ \}, \{b \leftarrow a;\ a\},\ b)$ *and* $Arg_2 = (\{\ \}, \{\ \}, \{a \leftarrow c;\ c\},\ a)$. *It is possible to rewrite* Arg_1 *into* $Arg_1{}' = (\{\ \}, \{\ \}, \{b \leftarrow a;\ a \leftarrow c;\ c\},\ b)$, *which is a non-conditional argument to* b.

Now consider $Arg_2{}' = (\{c\}, \{\ \}, \{a \leftarrow c;\ c\},\ a)$. *We can rewrite* Arg_1 *as* $Arg_1{}'' = (\{c\}, \{\ \}, \{b \leftarrow a;\ a \leftarrow c;\ c\},\ b)$, *which is a conditional argument with different conditions.*

The process of rewriting allows for agents to cooperate and build more elaborate arguments. In fact, any sequence of argument rewrites $Arg^0, Arg^1, \ldots, Arg^n$, provides a new argument, possibly conditional, that combines the knowledge of as many agents as the number of authors of arguments in that sequence. In particular, if the sequence ends with $Arg^n = (\emptyset, F^n, Hyp^n, G)$, the argument is eligible for being accepted in our framework.

EXAMPLE 6. *Consider the agents and the goal* $G_6 = a, b, d$ *taken from Example 1, together with the explanations below:*

$$Ag_1:\ Ex_6 = (E_6, F_6) = (\{b, d\}, \{\ \})\ for\ G_6.$$
$$Ag_2:\ Ex_3 = (E_3, F_3) = (\{d\}, \{\ \})\ for\ G_3 = b.$$

The agent Ag_1 *cannot satisfy* G_6 *but can build the explanation* Ex_6 *to it. The agent can build the conditional argument* $A_1 = (\{b, d\}, \{\ \}, \{b;\ a \leftarrow b;\ d\},\ G_6)$ *in* $P' = P \cup \{b, d\}$. *Then,* Ag_2 *should try to fulfill the conditions of* A_1, *with* $A_2 = (\{d\}, \{\ \}, \{d;\ b \leftarrow d;\},\ G_3)$ *to rewrite* Arg_1 *into* $Arg_1{}' = (\{d\}, \{\ \}, \{d;\ b \leftarrow d;\ a \leftarrow b;\},\ G_6)$, *which has less conditions. Finally,* Ag_3 *can complement this condition, since* $P_3 \models d$. *The third agent uses* $A_3 = (\{\ \}, \{\ \}, \{d\},\ d)$ *to rewrite* $Arg_1{}'$ *into* $Arg_1{}'' = (\{\ \}, \{\ \}, \{d;\ b \leftarrow d;\ a \leftarrow b;\},\ G_6)$, *which is a non-conditional argument.*

Arguments built with rewrites are possibly not derivable by any of the agents individually (only by the group), which is the case with $Arg_1{}''$ above. An important property of such arguments is that any attacks the agents can place against Arg_i, will also attack those obtained by rewriting Arg_i.

THEOREM 1. *If* $Arg_{i+1} = (E_{i+1}, F_{i+1}, Hyp_{i+1}, G_{i+1})$ *is a rewrite of* $Arg_i = (E_i, F_i, Hyp_i, G_i)$ *and there is an argument* $Arg_j = (E_j, F_j, Hyp_j, G_j)$ *that attacks* Arg_i, *then* Arg_j *is also an attack to* Arg_{i+1}.

PROOF. By exhaustion, suppose Arg_j is an attack that negates a literal or NAF-literal L in

G_i: Since $G_{i+1} = G_i$, Arg_j also attacks Arg_{i+1};

Hyp_i: Observe that the process of rewriting arguments will never remove hypotheses of Hyp_i that are program rules, except for facts. Because Hyp_i is minimal, there should be at least one rule $r \in Hyp_i$ with $L \in body(r)$. As this rule is still a hypothesis in Arg_{i+1}, we conclude that Arg_j is an attack to Arg_{i+1};

$C(E_i, F_i)$: Given that (E_i, F_i) is a minimal explanation, each and every condition is also in the body of at least one rule $r \in Hyp_i$ or in the goal G_i. Therefore, even if $L \notin C(E_{i+1}, F_{i+1})$, it is sure that $L \in body(r)$, for some $r \in Hyp_{i+1}$ or $L \in G_{i+1}$, so Arg_j attacks Arg_{i+1}.

Therefore, attacks are conserved over argument rewrites. □

3.3 The Role of Integrity Constraints

In a cooperative dialogue, the agents attack arguments they do not agree with, but also allow themselves to be convinced otherwise by their teammates. For that reason, the better an agent explains why it disagrees with an argument, the better that agent contributes to the collective goal building a group position towards the arguments played. Therefore, in such a cooperative setting, it makes sense for agents to share the integrity constraints in their knowledge bases whenever an argument would violate it. In that case, the agent will attack with a constraint-based argument.

DEFINITION 8. *An argument* $Arg_1 = (E_1, F_1, Hyp_1, G_1)$ *is a constraint-based attack to* $Arg_2 = (E_2, F_2, Hyp_2, G_2)$ *if*

1. $C(E_1, F_1) \subseteq Lit(Arg_2)$, *minimal w.r.t. set inclusion;*
2. $G_1 = \bot$ *(to express there is an inconsistency);*
3. *There is an integrity constraint* $r_1 \in Hyp_1$ *such that* $(Hyp_1 \setminus \{r_1\}) \cup C(E_1, F_1) \models L$, *for each* $L \in body(r_1)$.

If Hyp_1 *is unitary,* Arg_1 *is an* impossibility attack.

Constraint-based attacks enable agents to propose the rejection of an argument or goal to the group because it does not comply with some of the agents' integrity constraints.

Let r be an integrity constraint violated by Arg in $\langle P, H\rangle$, i.e., all literals in the body of r are true in Arg. To build a constraint-based attack, the agent reasons in the program $\langle P \setminus \{r\}, Lit(Arg)\rangle$ to build an $\overline{Arg} = (\overline{E}, \overline{F}, \overline{Hyp}, \overline{G})$ that justifies the goal $\overline{G} = \bigwedge\{L \mid L \in body(r)\}$. The argument $Arg' = (\overline{E}, \overline{F}, \overline{Hyp} \cup \{r\}, \bot)$ is a constraint-based attack.

An impossibility attack is an argument that is not subject to debate and puts an end to a sequence of arguments.

EXAMPLE 7. *Consider an agent Ag with knowledge base represented by the following ALP* $\langle P, H\rangle$*:*

$$\begin{array}{ll} P: & a \leftarrow b; \\ & c; \\ & \leftarrow a, c. \\ H: & \{b, c\} \end{array}$$

Consider $A_1 = (\{\ \}, \{\ \}, \{d \leftarrow a, c;\ a;\ c\},\ d)$*, an argument that violates the integrity constraint* $\leftarrow a, c$ *in* $\langle P, H\rangle$*. The agent Ag, then, attacks* A_1 *with the constraint-based attack* $(\{a\}, \{\ \}, \{c; \leftarrow a, c\}, \bot)$*. Now, consider* $A_2 = (\{\ \}, \{\ \}, \{a;\ c \leftarrow a\}, G)$*,* $G = a, c$*, that violates the integrity constraint* $\leftarrow a, c$ *in* $\langle P, H\rangle$*. The agent cannot accept it and produces the impossibility attack* $(\{a, c\}, \{\ \}, \{\leftarrow a, c\}, \bot)$*.*

We highlight that constraint-based attacks considers only the facts and conclusions from the argument as abducibles.

3.4 Evaluating Arguments Together

For a group of agents to accept an argument, it is necessary that all attacks against it had been proven inviable, so any attacks the argument can receive should be evaluated before it. As a consequence, the first arguments accepted will be those that receive no attacks or only received conditional attacks whose conditions could not be complemented by the group. After accepting an argument, the agents should check for consequences in the acceptance of other arguments in the dialogue: An accepted argument will disqualify the ones it attacks and arguments that cannot be further attacked will get accepted.

DEFINITION 9 (ACCEPTED ARGUMENTS). *An argument* $Arg_1 = (E, F, Hyp, G)$ *is accepted by an agent with knowledge base represented by the ALP* $\langle P, H\rangle$ *if* $E = \emptyset$ *(*F *do not need to be empty), and*

1. *there exists an answer set* S *of* P *with which* Arg_1 *is consistent, i.e.,* $S \cup Lit(Arg_1)$ *does not violates any integrity constraints in* P *and there is no* $L \in Lit(Arg_1)$ *such that* $neg(L) \in S$*; or*
2. *every attack the agent can place against* Arg_1 *is defeated by another argument accepted by it.*

A group of agents accepts an argument if all agents in the group accept it.

Let S be an answer set of P and consider a $L \in S$ such that $L = neg(L')$, for some $L' \in Lit(Arg)$. The agent can build an attack $Arg' = (E, F, Hyp, L)$ in P such that $E = F = \emptyset$, $Hyp \subseteq P$ and $Hyp \models L$. It is also possible that the agent can build conditional attacks against Arg. If the argument violates an integrity constraint in P, the agent should build a constraint-based attack (Section 3.3). In that case, the constraint-based attack should be played first, as it consists of an attack based in all of its answer sets.

The following theorem draws a connection between our work and *abstract argumentation* [8]. An argumentation framework is a pair $\langle S, \rho\rangle$, where S is a set of arguments and $(e, f) \in \rho$, $e, f \in S$, if e attacks f. One important concept is that of a conflict-free set of arguments, in which no argument attacks any other. An *stable extension* is a conflict-free set of arguments $S' \subseteq S$ such that every argument in $S \setminus S'$ is attacked by an element of S'. An argumentation framework might have zero, one or multiple stable extensions.

THEOREM 2. *If* S *is the set of arguments with* $E = \emptyset$ *played in a discussion, and* $\rho = \{(e, f) \in S \times S \mid e \text{ attacks } f\}$*, then the set Acc of arguments accepted by the group is a stable extension of the argumentation framework* $\langle S, \rho\rangle$*.*

PROOF. (Sketch) An argument can only be added to *Acc* if none of its attackers is accepted, so no other argument in *Acc* attacks it and the set is conflict-free. Furthermore, every other argument that could be accepted (with $E = \emptyset$), but is not in *Acc*, was only rejected because it is attacked by an argument in *Acc*. □

EXAMPLE 8. *Consider a group of agents engaged in a discussion on the goal* $M_0 = a, b, not\ c$*. Now suppose the arguments played are (in order):*

- $Arg_1 = (\{\ \}, \{c\}, \{a; b \leftarrow a;\}, M_0)$*;*
- $Arg_2 = (\{\ \}, \{\ \}, \{d; c \leftarrow d;\}, c)$ *attacks* Arg_1*;*
- $Arg_3 = (\{\ \}, \{\ \}, \{a; \neg d \leftarrow a;\}, \neg d)$ *attacks* Arg_2*;*
- $Arg_4 = (\{e\}, \{\ \}, \{e; d \leftarrow e;\}, d)$ *attacks* Arg_3*.*

Furthermore, suppose that the agents cannot build any other arguments in the dialogue. As a result, Arg_4 *is not accepted by any of the agents, since the condition e was not complemented. Because there are no other attacks against* Arg_3 *and it is non-conditional,* Arg_3 *gets accepted and disqualifies* Arg_2*. Since the only attack to* Arg_1 *was defeated, it gets accepted and so does* M_0*. Please note that an argument being accepted means every agent accepts it as in Definition 9. Also, note that* $\{Arg_1, Arg_3\}$ *is a stable extension of the argumentation framework involving only the arguments with* $E = \emptyset$ *and the attack relation between them.*

4. COOPERATIVE DIALOGUES

In this section, we investigate the acceptance of a goal or argument by a group of agents as we consider how agents deliberate individually and as a group. The satisfiability of a goal is debated as the agents place arguments to support it and others attack or rewrite them, possibly combining their knowledge in a cooperative process of group deliberation. We suppose the agents are willing to work their arguments for the best interest of the group and are honest. We also assume the agents have their knowledge bases built on the top of a common ontology and that they share the same language for communication.

The ultimate goal of the group is to build a group position towards a matter of discussion (a subject). To achieve that, the agents will take part in a dialogue, i.e., they will take turns playing arguments. The dialogue evolves through a succession of rounds in which every agent plays once, either making a move or passing. An agent will only pass if it evaluates an argument and accepts it or if it cannot play arguments. Every time a new argument is added, accepted or rejected by the group, the current matter of discussion is updated. In the first case, the recently added argument will be discussed next. If an argument is accepted or rejected, the agents will backtrack the dialogue to the last undecided matter, i.e., they will get back to further discuss the last subject that is still eligible for acceptance after receiving an attack. A single sequence of moves involving arguments in a dialogue is called a line of thought and the dialogue is the collection of lines of thought, which forms a tree with root on the initial matter of discussion.

4.1 Lines of Thought

DEFINITION 10. *A dialogue move is a quintuple* $Mv = (Arg, M, R, P, Agent)$ *where* Arg *is an argument,* M *is the matter of discussion at the time* Mv *is played, and* $R \in \{att, sup\}$ *indicates how the move is related to* M*, i.e., if it attacks (att) or supports (sup)* M*. Similarly,* $P \in \{T, F\}$ *is the position of the argument towards the initial matter of discussion* M_0 *being true (*T*) or false (*F*). If* M_0 *is a goal,* $P = T$ *(resp.* $P = F$*) means the goal can (resp. cannot) be satisfied. If* M_0 *is an argument,* $P = T$ *(resp.* $P = F$*) means the argument should be accept (resp. rejected) by the group. Finally,* $Agent$ *is the author of the move.*

The initial matter of discussion is represented by a different move Mv_0 that might present a goal instead of an argument. Either way, this move is not based in a matter of discussion ($M = NULL$) and supports itself ($R = sup$). It also suggests the initial matter of discussion is true ($P = T$), and has no author ($Agent = NULL$). Other moves are always played by agents. These attributes of each move are kept to assure consistent reasoning during the dialogue, as well as properly backtracking the dialogues as necessary.

The following definition resembles the concepts of *argument dialogues* and *argument dialogue trees* from [1].

DEFINITION 11. *In a cooperative dialogue with k agents, a* line of thought *on a matter* M_0 *is a nonempty finite sequence of moves* $Mv_i = (Arg_i, M_i, R_i, P_i, Agent_i)$*,* $i \geq 0$ *such that*

1. $Mv_0 = (M_0, NULL, sup, T, NULL)$.
2. *If* $i > 0$*,* Arg_i *is an argument,* M_i *is a matter,* $R_i \in \{att, sup\}$*,* $P_i \in \{T, F\}$ *and* $Agent_i \in \{Ag_1, \ldots, Ag_k\}$*;*
3. *For some agent* Ag_l*, if* M_0 *is a goal, the first move played is* $Mv_1 = (Arg_1, M_0, sup, T, Ag_l)$*, since* Arg_1 *should justify it. Otherwise, if* M_0 *is an argument,* $Mv_1 = (Arg_1, M_0, att, F, Ag_l)$ *and* Arg_1 *attacks* M_0*;*
4. $Agent_{i+1} \neq Agent_i$*;*
5. *For any* $i \neq j$*,* Arg_i *is a different argument from* Arg_j*;*
6. *If* Arg_{i+1} *attacks* Arg_i*, then* $R_{i+1} = att$ *and* $P_{i+1} \neq P_i$*; If* Arg_{i+1} *rewrites* Arg_i*, then* $R_{i+1} = sup$ *and* $P_{i+1} = P_i$*;*
7. *If two moves* Mv_i, Mv_j *have* $P_i = P_j$*, the arguments used are consistent towards one another, i.e., there is no pair* L*,* $neg(L)$ *in* $Lit(Arg_i) \cup Lit(Arg_j)$*.*
8. *If* Mv_i, Mv_j*,* $j > i$*, are moves in the same line of thought, then* Arg_j *is not attacked by* Arg_i*.*

A cooperative dialogue tree *is a finite tree with root in* Mv_0 *and where each branch is a line of thought. In such a tree, if two moves* Mv_j, Mv_k *are played after the same* Mv_i *in different lines of thought (a ramification), then* $Arg_j \neq Arg_k$*.*

A cooperative dialogue is developed as different lines of thought are explored by the agents. When an argument is played, it starts the process undecided (neither accepted nor rejected) as it might be attacked, so that argument becomes the current matter of discussion. If no agents will rewrite or attack the current matter, the line of thought reaches its end and that last argument is evaluated. The agents will then reconsider the previously played arguments in that line of thought in reverse order (backtrack), evaluating or further attacking/rewriting matters as possible. A dialogue stops when the first argument in a line of thought (other than M_0) is accepted. In that case, the initial matter gets satisfied (if it is a goal) or rejected (if it is an argument). Alternatively, the dialogue ends if the current matter is M_0, but no moves can be made to develop new lines of thought. In that case, the group can not satisfy the initial matter (if it is a goal) or has to accept it (if it is an argument).

To avoid repeating parts of the dialogue and assure consistent reasoning, the group keeps record of the sets of arguments accepted (Acc) and rejected (Rej). These sets are initially empty and are updated as arguments are evaluated by the group. We use the symbol $\Leftarrow$ to express updates.

When the group concludes the evaluation of an argument Arg_i (played in the move Mv_i):

- If Arg_i is accepted by the group, $Acc \Leftarrow Acc \cup \{Arg_i\}$. The group backtracks the dialogue to the last move $Mv_j = (Arg_j, M_j, R_j, P_j, Agent_j)$ with $P_j \neq P_i$ in the same line of thought and rejects Arg_j (see below). If no such Mv_j exists, the argument has the same position as the initial matter, so it is a goal that gets satisfied and the dialogue is finished.
- If Arg_i is rejected by the group, $Rej \Leftarrow Rej \cup \{Arg_i\}$. The group backtracks the dialogue to the previous movement in the same line of thought and continues the dialogue. If no such movement exists, then Arg_i is the initial matter, so the dialogue ends.

The dialogue tree and the sets Acc, Rej are kept accessible to all agents (as a blackboard). An agent will only play an argument Arg_j if it can still be accepted by the group, i.e., no arguments in Acc attack Arg_j at the time it is played.

PROPOSITION 1. *Every line of thought is finite, and so is the dialogue tree.*

PROOF. (sketch) Arguments cannot be repeated in the same line of thought. Therefore, attacks and rewrites are limited and a line of thought cannot be infinite since no cycles appear. Also, the language of the agents is finite and different lines of thought have to start with different arguments, so the dialogue tree is also finite. □

Please note that each line of thought and the sets Acc and Rej grow monotonically, since new arguments are introduced, but none is removed. In addition, our concept of line of thought assures the existence a stable extension (possibly more than one) over the arguments in the dialogue, as stated in Theorem 2. This is a consequence of our restrictions on what kinds of arguments can be played. Such restrictions also assure that, given a subject for discussion, the agents exhibit consistent group reasoning over two opposite positions and no argument is left undecided.

PROPOSITION 2. *The dialogue tree is developed as a depth-first search for a set of arguments accepted by the group that defines the group position towards the initial matter.*

PROOF. (sketch) The agents will always consider the last argument played to produce moves in the dialogue, and an argument is evaluated when no attacks to it can be played. That way, a single line of thought is explored at a time and possible ramifications are only considered while backtracking the arguments in a line of thought. □

4.2 Individual Deliberation

In a group of agents deliberating cooperatively, the parties propose arguments and collectively study the possible flaws these might have. In order to do so, agents will take turns to play arguments as they reason over two opposing positions: One that supports the initial matter (acceptance of an argument or satisfaction of a goal) and another that is against it. All agents should argue over both positions in an attempt to better explore the combination of their knowledge bases. In what follows, when we say that an agent tries to build or searches for an argument, we mean an argument that can still be played, i.e, that is not defeated by arguments previously accepted by the group.

In each turn of an agent, it will conceive available moves to play in the current line of thought. To do that, the agent considers the current matter of discussion M and deliberates accordingly by attempting the following steps (in order):

- If M is a goal G:
  1. build an argument in P to justify G.
  2. build a conditional argument to justify G.
- If M is an argument $Arg = (E, F, Hyp, G)$:
  1. verify if there is an $A \in Acc$ that disqualifies Arg;
  2. accept Arg, i.e., verify if it is accepted;
  3. build a constraint-based attack (Def. 8) to Arg;
  4. build an argument in P to attack Arg;
  5. build a conditional argument to attack Arg;
  6. build an argument in P to rewrite Arg;
  7. build a conditional argument to rewrite Arg;

In each case, if the agent succeeds in a step, it will play the argument built (if this is the case) and finish its turn without trying the others. If M is an argument Arg and the agent accepts it, the agent will pass its turn without making a move. In case an agent fails in all steps, it will also pass, for it cannot make a move.

If the current matter of discussion is a goal, the agent attempts to build an argument to justify it. If it is an argument Arg, but it is not consistent with an answer set of P, the agent will try to attack it. In that case, the agent will first attempt constraint-based attacks, since violating an integrity constraint means the argument might be inconsistent with multiple answer sets. Next, the agent tries to build an attack (non-conditional) in P, based on an answer set. Finally, the agent appeals to explanations and conditional arguments to disqualify the argument in question. If the argument is conditional with $E \neq \emptyset$, the agent should try to rewrite the argument. If no attacks can be played and the argument has $E = \emptyset$, the agent has to accept it (Definition 9), so it passes its turn (no moves available).

4.3 Dialogue Example

Our dialogue framework proposes a model for group deliberation that is fair as all agents have the same number of chances to play arguments in the discussion. In this process, agents cannot only state their individual arguments and demands on each matter of discussion, but also combine their knowledge to build collective arguments. These agents take turns placing arguments and might get convinced by their colleagues to accept opinions they would not if they were on their own. Next, we show an example of dialogue with two lines of thought. For an easier comprehension of the example, we will only show the arguments involved in each move. We will show the updates of Acc, Rej and the current matter of discussion.

EXAMPLE 9. *Consider a group of three agents engage in a discussion on the matter $M_0 = a, b, not\ c$. In the sequel, we will list the arguments placed by the agents and write (round, turn) to enumerate them.*

(1, 1) Ag_1 plays $Arg_1 = (\{\}, \{c\}, \{a; b \leftarrow a;\}, M_0)$;

(1, 2) Ag_2 attacks Arg_1 with $Arg_2 = (\{\}, \{\}, \{d; c \leftarrow d;\}, c)$;

(1, 3) Ag_3 attacks Arg_2 with $Arg_3 = (\{\}, \{\}, \{a; \neg d \leftarrow a;\}, \neg d)$;

(2, 1) Ag_1 accepts Arg_3 and passes;

(2, 2) Ag_2 attacks Arg_3 with $Arg_4 = (\{e\}, \{\}, \{e; d \leftarrow e;\}, d)$;

(2, 3) Ag_3 cannot attack or rewrite Arg_4 (pass);

(3, 1) Ag_1 cannot attack or rewrite Arg_4 (pass). $Rej \Leftarrow Rej \cup \{Arg_4\}$, $M \Leftarrow Arg_3$;

(3, 2) Ag_2 attacks Arg_3 with $Arg_5 = (\{\}, \{\}, \{\neg a \leftarrow not\ a;\}, \neg a)$;

(3, 3) Ag_3 attacks Arg_5 with $Arg_6 = (\{\}, \{\}, \{a;\}, a)$;

(4, 1) Ag_1 accepts Arg_6 and passes;

(4, 2) Ag_2 cannot attack or rewrite Arg_6 (pass). $Acc \Leftarrow Acc \cup \{Arg_6\}$, $Rej \Leftarrow Rej \cup \{Arg_5\}$, $M \Leftarrow Arg_3$;

(4, 3) Ag_3 accepts its own argument Arg_3 (pass);

(5, 1) Ag_1 accepts Arg_3 and passes;

(5, 2) Ag_2 cannot attack or rewrite Arg_3 (pass). $Acc \Leftarrow Acc \cup \{Arg_3\}$, $Rej \Leftarrow Rej \cup \{Arg_2\}$, $M \Leftarrow Arg_1$;

(5, 3) Ag_3 accepts Arg_1 (pass);

(6, 1) Ag_1 accepts Arg_1 (pass);

(6, 2) Ag_2 accepts Arg_1, for it cannot attack or rewrite it, and passes its turn. $Acc \Leftarrow Acc \cup \{Arg_1\}$.

As a result of the acceptance of Arg_1, the initial matter M_0 is also accepted and the discussion is finished.

In each step of the dialogue, the current matter is updated to the last argument placed (after a move) or left undecided (after arguments get evaluated). An argument is evaluated if a full round passes and the agents do not make any moves. In Example 9, if Ag_2 were able to attack Arg_3 on step (5,2) or Arg_1 on step (6,2), the dialogue would continue on a different line of thought. Please note that Ag_2 cannot use Arg_5 to attack the arguments Arg_3 and Arg_1 in different lines of thought because it has been rejected in step (4,2).

5. RELATED WORK

Argumentative Deliberation [13] involves the use of arguments by agents to support self deliberation and also employs abductive reasoning. This approach introduces conditional arguments that are played in a dialogue, but the abductive hypotheses are not shared as such. In a cooperative setting, however, it makes sense to share them with other agents. In our work, a group of agents can share hypothesis to combine their knowledge and produce interesting arguments that they would possibly not be able to conceive individually. Judgement Aggregation [4] allows agents in a group to combine their individual judgements over a set of arguments and collectively decide which ones to accept. Unlike our work, this approach does not consider communication amongst the agents. Abductive reasoning and argumentation have also been combined together in [2, 15]. In

both papers, the explanations and arguments are produced by a single agent at a time and their knowledge is not combined. In our work, agents share their hypothesis to combine their knowledge and reach consensus over acceptable arguments and a group position towards a matter of discussion. In [9, 5, 3] agents can share hypotheses to produce group explanations. In our proposal, agents will provide arguments to support or attack each others hypotheses. Our goal, however, is not to produce group explanations with combined hypotheses, but for agents to point out missing pieces of their arguments, which can be complemented or criticized by others. The works in [6, 18] study collaborations in distributed argumentation as agents form coalitions to produce group arguments. They consider partial arguments, which are partial derivations of arguments that need complimentary knowledge. Althought unclaimed, the kind of reasoning they introduce is clearly abductive. Our work innovates as group arguments are built in a dialogue and extensively discussed by the agents, so the conditional arguments, which are much similar to partial arguments, might also be attacked and are subject to rejection. Another important difference is that our agents are able to detect possible inconsistencies amongst their beliefs as we recur to extended abduction [16, 17].

6. CONCLUSION AND FUTURE WORKS

We presented an approach to cooperative dialogues in groups of agents. In our framework, agents can play arguments and attack the opinions of each other, but can also complement them and build more elaborate ones. This innovative feature allows agents to rewrite arguments and combine their knowledge as they search for an unified group opinion about some matter of discussion and their own arguments. We have enabled such a cooperative behavior by employing abductive reasoning and changing the way that abduction-based conditional arguments are placed in a dialogue. This cooperative behavior can be also perceived as group deliberation. Even though we only consider arguments and goals as initial matters of discussion, a group can deliberate about how to accomplish agent goals, evaluate proposals in a negotiation, make group decisions, and so on, giving our framework a number of different applications. In our future works, we will explore these applications, their particularities, and study what kinds of roles the individual preferences of agents should play in the process.

7. ACKNOWLEDGEMENTS

Research partially supported by CAPES (PROCAD).

8. REFERENCES

[1] L. Amgoud, S. Parsons, and N. Maudet. Arguments, dialogue, and negotiation. In W. Horn, editor, *ECAI*, pages 338–342. IOS Press, 2000.

[2] F. Bex and H. Prakken. Investigating stories in a formal dialogue game. In *Proceeding of the Conference on Computational Models of Argument (COMMA 2008)*, pages 73–84. IOS Press, 2008.

[3] G. Bourgne, K. Inoue, and N. Maudet. Abduction of distributed theories through local interactions. In H. Coelho, R. Studer, and M. Wooldridge, editors, *ECAI*, volume 215 of *Frontiers in Artif. Intell*, pages 901–906. IOS Press, 2010.

[4] M. Caminada and G. Pigozzi. On judgment aggregation in abstract argumentation. *Autonomous Agents and Multi-Agent Systems*, 22(1):64–102, 2011.

[5] A. Ciampolini, E. Lamma, P. Mello, F. Toni, and P. Torroni. Cooperation and competition in alias: A logic framework for agents that negotiate. *Ann. Math. Artif. Intell.*, 37(1-2):65–91, 2003.

[6] I. de Almeida Móra, J. J. Alferes, and M. Schroeder. Argumentation and cooperation for distributed extended logic programs. In *Nonmonotonic Reasoning Workshop*, 1998.

[7] M. Denecker and A. C. Kakas. Abduction in logic programming. In A. C. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2407 of *LNCS*, pages 402–436. Springer, 2002.

[8] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artif. Intell.*, 77(2):321–358, 1995.

[9] M. Gavanelli, E. Lamma, P. Mello, and P. Torroni. An abductive framework for information exchange in multi-agent systems. In J. Dix and J. A. Leite, editors, *CLIMA*, LNCS, vol. 3259, pages 34–52. Springer, 2004.

[10] M. Gelfond and V. Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Comput.*, 9(3/4):365–386, 1991.

[11] K. Inoue and C. Sakama. Abductive framework for nonmonotonic theory change. In *Proc. 14th International Joint Conference on Artificial Intelligence - Volume 1*, pages 204–210, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc.

[12] A. C. Kakas, R. A. Kowalski, and F. Toni. The role of abduction in logic programming. In D. Gabbay, C. Hogger, and J. Robinson, editors, *Handbook in Artificial Intelligence and Logic Programming*, volume 5 of *Datalogiske Skrifter*, pages 235–324. Roskilde University, 1998.

[13] A. C. Kakas and P. Moraitis. Argumentative agent deliberation, roles and context. In J. Dix, J. A. Leite, and K. Satoh, editors, *CLIMA*, Datalogiske Skrifter, vol. 93, pages 35–48. Roskilde University, 2002.

[14] P. McBurney, D. Hitchcock, and S. Parsons. The eightfold way of deliberation dialogue. *Int. J. Intell. Syst.*, 22(1):95–132, 2007.

[15] F. Sadri, F. Toni, and P. Torroni. Logic agents, dialogues and negotiation: An abductive approach. In *In Proc. AISB'01 Convention*. AISB, 2001.

[16] C. Sakama and K. Inoue. An abductive framework for computing knowledge base updates. *Theory Pract. Log. Program.*, 3:671–715, November 2003.

[17] C. Sakama and K. Inoue. Negotiation by abduction and relaxation. In *Proc. 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS'07, pages 242:1–242:8, New York, NY, USA, 2007. ACM.

[18] M. Thimm, A. J. Garcia, G. Kern-Isberner, and G. R. Simari. Using collaborations for distributed argumentation with defeasible logic programming. In M. Pagnucco and M. Thielscher, editors, *NMR'08*, pages 179–188. University of New South Wales, Technical Report No. UNSW-CSE-TR-0819, September 2008.

Defeasible Argumentation for Multi-Agent Planning in Ambient Intelligence Applications

Sergio Pajares Ferrando
Dpto. de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de Vera, s/n, 46022 Valencia, Spain
spajares@dsic.upv.es

Eva Onaindia
Dpto. de Sistemas Informáticos y Computación
Universitat Politècnica de València
Camino de Vera, s/n, 46022 Valencia, Spain
onaindia@dsic.upv.es

ABSTRACT

This contribution presents a practical extension of a theoretical model for multi-agent planning based upon DeLP, an argumentation-based defeasible logic. Our framework, named DeLP-MAPOP, is implemented on a platform for open multi-agent systems and has been experimentally tested, among others, in applications of ambient intelligence in the field of health-care. DeLP-MAPOP is based on a multi-agent partial order planning paradigm in which agents have diverse abilities, use an argumentation-based defeasible reasoning to support their own beliefs and refute the beliefs of the others according to their knowledge during the plan search process. The requirements of Ambient Intelligence (AmI) environments featured by the imperfect nature of the context information and heterogeneity of the involved agents make defeasible argumentation be an ideal approach to resolve potential conflicts caused by the contradictory information coming from the ambient agents. Moreover, the ability of AmI systems to build a course of action to achieve the user's needs is also a claiming capability in such systems. DeLP-MAPOP shows to be an adequate approach to tackle AmI problems as it gathers together in a single framework the ability of planning while it allows agents to put forward arguments that support or argue upon the accuracy, unambiguity and reliability of the context-aware information.

Categories and Subject Descriptors

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

General Terms

Algorithms, Experimentation

Keywords

Defeasible Argumentation, Multi-Agent Planning, Ambient Intelligence.

1. INTRODUCTION

Ambient Intelligence (AmI) integrates concepts ranging from Ubiquitous Computing to Artificial Intelligence with the vision that technology will become invisible, embedded in our natural surroundings, present whenever we need it, and adaptive to users [1].

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

In AmI environments, people are surrounded with networks of embedded intelligent devices that can sense the available context information, anticipate, and perhaps adapt to their needs. In this contribution, we handle these requirements by modeling ambient agents as entities which manage a portion of the AmI environment, i.e. they are responsible for one or more devices. Due to the imperfect nature of the context and the heterogeneity of ambient agents, whose different viewpoints lead them to infer different assumptions about the user's current situation, ambient agents, as distributed autonomous software entities, are required to engage in interactions, argue with one another, and make agreements, individually or collectively, while responding to changing circumstances of the ambient environment. For this reason, ambient agents are being advocated as a next-generation model for engineering complex distributed systems such as AmI systems. The aim in AmI is to make the interaction between users and the smart environment easy.

Defeasible is the opposite of irrefutable or indisputable. A defeasible piece of information is a non-demonstrative piece of information that is acknowledged to be able to fail or be corrected. Defeasible reasoning is usually realized as a rule-based approach for reasoning with incomplete and inconsistent information through the use of rules that may be defeated by other rules. Defeasible reasoning has been successfully used in AmI applications [2]. On the other hand, **Defeasible Argumentation**, which has recently become a very active research field in computer science [3], is a form of defeasible reasoning that emphasizes the notion of argument. An argument is a chain of reasoning that concludes one piece of information (conclusion) on the basis of some other pieces of information (premises). Thus, defeasible argumentation can be viewed as a powerful tool for reasoning about inconsistent information through a rational interaction of arguments for and against some conclusion derived by an ambient agent. Defeasible argumentation has also been successfully proved in AmI applications [4].

The defeasible logic programming formalism DeLP [5] is one of the most popular approaches to build defeasible argumentation. Our framework, DeLP-MAPOP, builds upon DeLP to implement the defeasible argumentation mechanism. The key element of DeLP are defeasible rules (Head $\prec$ Body), which are used to represent a deductive relation between pieces of knowledge that could be defeated once another piece of knowledge is considered. For instance, a defeasible rule like *emergency* $\prec$ *patient-fever* denotes that an ambient agent believes that if the monitoring system returns the patient has fever then there are provable reasons to declare an emergency. The defeasible rule $\sim$*emergency* $\prec$ *{normal-pulse, conscious, correct-breathing}* provides reasons to believe the contrary, in whose case we say that the first piece of information is acknowledged to fail in case *{normal-pulse, conscious, correct-breathing}* hold in the context. However, assuming that

another ambient agent knows that the patient is vomiting blood, i.e. {*bloody-vomit*} holds in the context, then it might derive the patient has not a normal pulse by following the defeasible rules {$\sim$*normal-pulse* $\prec$ *internal-bleeding*; *internal-bleeding* $\prec$ *bloody-vomit*}, which represents an attack to the defeasible rule whose conclusion is $\sim$*emergency*. Thus, arguments (combinations of defeasible rules and facts) for conflicting pieces of information are built, and then compared to decide which one prevails.

Planning is a desired ability in AmI systems to achieve a goal-oriented behavior, i.e. to decide the course of action to meet the needs of the specific application, for instance, stabilizing a patient in a home-care system. Planning has been used in some AmI applications for monitoring and responding to the needs of a diabetic patient [6]. Particularly, the work in [6] presents a centralized planner that manages distributed capabilities as it assumes that some agents do not have planning capabilities. In this case, an agent is implemented as a device, which prevents the agent from taking responsibilities in building the plan due to its limitations in processing and communication; for example, a cell phone could not be able to autonomously plan to call a doctor given that other devices detected that a user in the environment is ill [6]. However, in our contribution an ambient agent is executed on an independent host and can encompass several devices. This increases the communication capacity as well as autonomy and endow agents with the necessary abilities to pose a goal and build a plan for this goal. This approach allows us to address many real applications where the capabilities to perceive the context and perform the actions are distributed across agents. **Multi-Agent Planning** (MAP) applied to an AmI environment is intended as the ability of a team of ambient agents to build collaboratively a plan of actions that, when performed in the AmI context, meets the needs and goals of the application.

Partial Order Planning (POP) is a suitable planning approach to address the requirements derived from a distributed planning thanks to the application of a non-sequential behaviour and the least commitment principle [7]. This is evidenced by the fact that most existing architectures for integrating planning with execution, information gathering and scheduling are based on partial order planners. In [8], authors argue that POP-based frameworks offer a more promising approach for handling domains with durative actions and temporal and resource constraints as compared to other planning approaches. In fact, most of the known implementations of planning systems capable of handling temporal and durative constraints (e.g. NASA's RAX [9]) are based on the POP paradigm. Even for simple planning tasks, partial order planners offer a higher degree of execution flexibility. For these reasons, this work is based on Multi-Agent Partial Order Planning (MAPOP).

A extension of POP with DeLP-style argumentation, denoted as DeLP-POP, was introduced in [10], where both actions and arguments may be used to enforce some goal, if their conditions (are known to) apply and arguments are not defeated by other arguments applying. Unlike actions, arguments are not only introduced to intentionally support some step of a plan, but they are also presented to defeat or defend other supporting arguments in the plan. When actions and arguments are combined in a partial order plan, new types of interferences or threats appear [10] which need to be identified and resolved to obtain valid plans. Finally, the work in [11, 12, 13] proposes an extension of the DeLP-POP to a multi-agent environment. Specifically, it proposes a dialogue for argumentative plan search, by which agents exchange plan proposals and arguments for or against such proposals. To the best of our knowledge, these theoretical works have neither been implemented nor tested on real-world domains such as AmI applications.

This contribution presents DeLP-MAPOP, a system that combines, implements and tests features like multi-agent defeasible argumentation and multi-agent planning in AmI applications. DeLP-MAPOP develops and implements an extended and refined version of the framework presented in [12]; DeLP-MAPOP is applied and experimentally tested in an AmI environment, it extends the agents' knowledge bases and the dialogues during the plan search and it offers a new classification of planning interferences. The remainder of this paper is divided as follows. First, we introduce the basic elements of the system; then we present the MAP protocol applied in an AmI scenario to deal with a person suffering from a heart disease. Next, the experiments carried out to validate the present work are described and analyzed. Finally, we conclude and present some directions for future work.

2. COMPONENTS OF THE SYSTEM

In this section, we provide definitions for the notions of ambient agent, context information, planning task, argument versus action and plan, that will be later used for the definition of the DeLP-MAPOP protocol.

2.1 Ambient Agents

In DeLP-MAPOP, ambient agents act as planning agents with different beliefs, capabilities and preferences. Thus, we assume the capabilities to perceive the context, perform actions and derive new conclusions are distributed across ambient agents. Agents are managed and supervised by the Agent Management System (AMS) that is responsible for the following tasks: i) Exercising supervisory control over access to the multi-agent platform; it is responsible for authentication of resident ambient agents and control of registrations. ii) Discovering new user's needs generated directly by the user or indirectly by a smart device, which provides the input to a DeLP-MAPOP process in terms of goals to be reached. iii) When ii) occurs, the AMS agent gathers the ambient agents who will participate in the planning process and will return the action plan to satisfy the user's needs. For instance, a device that monitors the patient's heart's rate may detect the presence of arrhythmias by means of an electrocardiogram, a symptom that might entail a heart attack. In this case, the monitoring system generates the goal *patient-to-be-treated*, and communicates it to the AMS agent.

The knowledge of an ambient agent mainly comprises context information encoded as defeasible rules and initial facts, and context capabilities represented as planning actions.

2.2 Context information

The representation scheme used by DeLP-MAPOP to model components of the AmI environment is based on a state-variable representation, where variables map to a finite domain of values which represent the problem objects. A state-variable representation is equivalent to a classical planning representation in expressive power and it is also useful in non-classical planning problems as a way to handle numbers, functions and time. In this paper, we will restrict our attention to only non-numeric variables. Since actions change the state of the world and defeasible rules make assumptions about the state of the world, actions and defeasible rules are most naturally modeled as elements that change the values of the state variables. The variable-value pair $\langle v_i, vl_i \rangle$ denotes the value vl_i is assigned to the variable v_i. For instance, the variable-value pair $\langle$ *at-amb, pH* $\rangle$ indicates that the ambulance amb is located at the patient's home pH, that is, the value of the variable denoting the position of the ambulance is the patient's home.

In what follows, we define the set of elements used to represent the agent's context information. (i) the set of objects O that model

the elements of the planning domain over which the actions and defeasible rules can act. (ii) the set of state variables V that are used to model the states of the world: each state variable $v_i \in V$ is mapped to a finite domain of mutually exclusive values D_{v_i}, where $\forall v_i \in V, D_{v_i} \subseteq O$. (iii) the initial state of the problem Ψ, which is a consistent set of variable-value pairs; a variable with no assigned value in the initial state is assumed to have an *unknown* value. (iv) the set of defeasible rules Δ, where each rule δ follows the form $\langle \mathsf{head}(\delta) \prec \mathsf{body}(\delta) \rangle$; if the set of variable-value pairs in $\mathsf{body}(\delta)$ is warranted, i.e. if variables have the specified values in the pair, then δ is applicable and for each $\langle v_i, vl_i \rangle$ that appears in the head of the rule, v_i is assigned the value vl_i. (v) A is the set of planning actions $\alpha = \langle \mathsf{P}(\alpha), \mathsf{X}(\alpha) \rangle$, where $\mathsf{P}(\alpha)$ is a set of preconditions encoded as variable-value pairs that must be satisfied in order to apply the effects in $\mathsf{X}(\alpha)$, also encoded as $\langle v_i, vl_i \rangle$.

2.3 Planning task

Each ambient agent $x \in \{\mathsf{Ag}_1 \dots \mathsf{Ag}_n\}$ is initially endowed with **a planning task** $\mathbb{M}_x = (O_x, V_x, \Psi_x, \Delta_x, A_x, F_x, G)$ where:

1. O_x is the set of objects known by the agent x.
2. V_x is the set of variables managed by agent x to represent the agent's knowledge about the state of the world.
3. $\Psi_x = \{\langle v_i, vl_i \rangle \mid v_i \in V_x; vl_i \in D_{v_i}\}$ represents the partial view of the initial world state of agent x, i.e. the information that agent x knows about the initial state. We assume $\bigcup_{x \in \{\mathsf{Ag}_1 \dots \mathsf{Ag}_n\}} \Psi_x$ is a consistent set.
4. Δ_x is a set of defeasible rules known by the agent x.
5. A_x is a set of planning actions known by the agent x.
6. F_x represents a consistent set of the agent-specific preferences $F_x \subseteq \{(a, d) \mid (a \in A_x),\ d \in [0, 100]\}$, where action a is preferred with the estimated interest degree d.
7. G is the set of global goals that represent the needs of a user in an AmI environment. G is expressed as a set of pairs variable-value thus indicating the value each variable is expected to assume in the final state. Unlike the rest of elements, G is known by all of the ambient agents.

2.4 Arguments versus Actions

As we saw in the Introduction section, and based on the framework presented in [10], both actions and arguments may be used to enforce some task goal in DeLP-MAPOP. As illustrated in Figure 1 (a), an **argument** $\mathcal{A}$ for $\langle v_i, vl_i \rangle$ proposed by an ambient agent Ag_1, is denoted as $\mathcal{A}^{\mathsf{Ag}_1} = (\{\mathsf{concl}(\mathcal{A}^{\mathsf{Ag}_1})\}, \{\mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1})\})$, where $\mathsf{concl}(\mathcal{A}^{\mathsf{Ag}_1}) = \langle v_i, vl_i \rangle$ is the argument conclusion and $\mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1})$ is a subset of defeasible rules such that $\mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1}) \subseteq \Delta_x$. $\mathcal{A}^{\mathsf{Ag}_1}$ is consistent if there exists a defeasible derivation for $\langle v_i, vl_i \rangle$ from $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1}) \cup \mathsf{rules}(\mathcal{A}^{\mathsf{Ag}_1})$, where $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$ is the argument base, the set of <variable,value> that must be warranted in the agent's context information. The existence of an argument $\mathcal{A}^{\mathsf{Ag}_1}$ does not suffice to warrant its conclusion $\langle v_i, vl_i \rangle$, this depends on the interactions among arguments as we will see in Section 3.3. We semantically distinguish between **supporting arguments** (also known as argument steps) as the arguments specifically used to support some goal of the plan, and **attacking arguments** (also known as defeaters) which are only introduced to attack some argument step previously introduced in the plan.

The difference between assigning a value to a variable by an argument or by an action is that in the case of a planning action the value is indisputable because it reflects a modification stated in the problem domain modelling; however, the confirmation of a value assigned to a variable by an argument depends on the interaction with other attacking arguments.

2.5 Plans

In POP, a partial order plan Π is a set of partially ordered actions (denoted by the relation $\prec$) which actually encodes multiple linear plans. More specifically, a **plan** Π is a tuple $\Pi = (A(\Pi), \mathcal{AR}(\Pi), G(\Pi), \mathcal{OC}(\Pi), \mathcal{CL}(\Pi), \mathcal{SL}(\Pi))$, where $A(\Pi)$ denotes the set of action steps, $\mathcal{AR}(\Pi)$ represents the set of argument steps, $G(\Pi)$ is the task's common goals (the user's needs), $\mathcal{OC}(\Pi)$ is a set of ordering constraints, and $\mathcal{CL}(\Pi)$ and $\mathcal{SL}(\Pi)$ represent the sets of causal and support links, respectively. In POP, Ψ and G are encoded as dummy actions $\{\alpha_\Psi \prec \alpha_G\}$ where α_Ψ is also refereed to as the initial step of the plan and α_G to as the final step of the plan, with $\mathsf{X}(\alpha_\Psi) = \Psi$, $\mathsf{P}(\alpha_G) = G$, and $\mathsf{P}(\alpha_\Psi) = \mathsf{X}(\alpha_G) = \emptyset$.

Let $\langle v_i, vl_i \rangle$ be an open goal in Figure 1(b), motivated by some action step $\alpha_G \in A(\Pi)$, i.e. $\langle v_i, vl_i \rangle \in \mathsf{P}(\alpha_G)$; let $\langle v_k, vl_k \rangle$ be another open goal, motivated by some argument step $\mathcal{A}^{\mathsf{Ag}_1}$, i.e. $\langle v_k, vl_k \rangle \in \mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$. Then, the goal $\langle v_i, vl_i \rangle \in \mathsf{P}(\alpha_G)$ must be supported by an argument, argument $\mathcal{A}^{\mathsf{Ag}_1}$ in Figure 1(b), which introduces a **support link** $(\mathcal{A}^{\mathsf{Ag}_1}, \langle v_i, vl_i \rangle, \alpha_G) \in \mathcal{SL}(\Pi)$, where $\mathcal{SL}(\Pi) \subseteq \Delta \times G(\Pi) \times A$. In contrast, the goal $\langle v_k, vl_k \rangle$ must be supported by an action, α_1 in Figure 1(b), which introduces a **causal link** $(\alpha_1, \langle v_k, vl_k \rangle, \mathcal{A}^{\mathsf{Ag}_1}) \in \mathcal{CL}(\Pi)$, where $\mathcal{CL}(\Pi) \subseteq A \times G(\Pi) \times \Delta$. Triangles in Figure 1(b) represent argument steps (i.e. arguments that support preconditions of action steps), while rectangles represent action steps (i.e. actions that support the basis of an argument step). Therefore, in this approach, goals must always be initially derived by some argument step, and an argument base must be satisfied by another action step (including the initial step). This way, a typical causal link in POP is now replaced by a causal link and a support link. Note this representation allows us to implicitly address *the qualification problem* [14] as every precondition of a planning action is now supported by an argument step rather than directly by an action effect. This way, agents may attack the fulfillment of such precondition if they believe that there exist other non-explicit conditions that prevent the supporting action from having its intended effects. This new conception of mandatorily supporting preconditions through argument steps gives rise to a new and unique notion of threat. Under this new perspective, the concept of argument-argument threat in [10, 12] is now replaced by a broader notion of argument-argument threat that covers all the interferences that arise between the elements of a plan in which the qualification problem is addressed through the use of argument steps. Depending on where these argument-argument threats occur in the plan, we will distinguish between **threats** (Section 3.2) and **attacks** (Section 3.3).

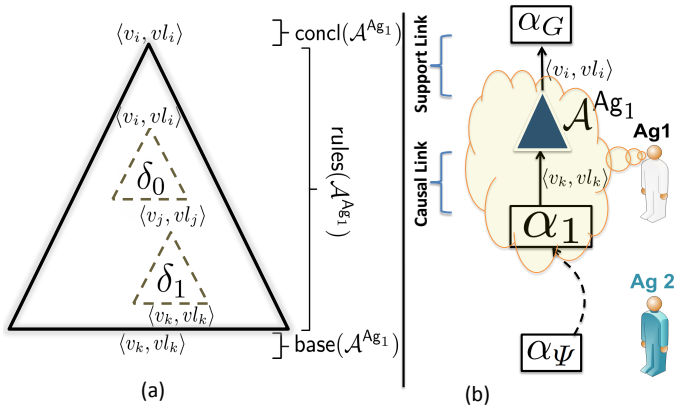


Figure 1: (a) An argument $\mathcal{A}^{\mathsf{Ag}_1}$ for $\langle v_i, vl_i \rangle$ by using two defeasible rules: $\delta_0 = \{\langle v_i, vl_i \rangle\} \prec \{\langle v_j, vl_j \rangle\}$ and $\delta_1 = \{\langle v_j, vl_j \rangle\} \prec \{\langle v_k, vl_k \rangle\}$, such that $v_i \neq v_j$ and $v_j \neq v_k$ and $\{v_i, v_j, v_k\} \subseteq V_x$; (b) An example of a partial plan.

3. MULTI-AGENT PLANNING PROTOCOL

First, we outline the procedure followed by the DeLP-MAPOP protocol that interleaves a planning stage, an argumentation stage and a selection stage. Given a set of global goals, G, that address the requirements of an AmI application, agents build their own planning task $\mathbb{M}_x$ so they can differently contribute to the construction of the joint solution plan. The starting point of the MAP protocol is an empty initial plan Π_0 and the output is the solution plan. Once checked the plan Π is not a solution, the first step is to select an open goal $\Phi \in G(\Pi)$ of the planning task for resolution (choose[1] step in Algorithm 1). Then it comes the planning stage (PROPOSALS step in Algorithm 1) where agents put forward and exchange different partial order plans that would potentially solve Φ. Following, agents get involved in an argumentative dialogue (EVALUATION step in Algorithm 1) in which they expose their arguments for or against each of the proposals. This evaluation process performs a *warranty procedure* to determine which proposals do not receive attacks or, otherwise, the received attacks do not succeed. Subsequently, ambient agents reach an agreement as to which about the next partial plan and they continue the search exploration (SELECTION step in Algorithm 1). The process is repeated until a solution plan is found.

Algorithm 1: Multi-agent planning protocol overview.

input : The initial plan $\Pi_0 := \{\alpha_\Psi \prec \alpha_G\}$.
output: The solution plan Π.

$\Pi := \Pi_0$
while $\Pi <> null$ **do**
  if $G(\Pi) = \emptyset$ **then**
    return Π *[It is a plan solution.]*
  else
    choose $\Phi \in G(\Pi)$;
    $\mathsf{Ref}(\Pi, \Phi) :=$ PROPOSALS(Π, Φ);
    [Each plan Π_r of the set $\mathsf{Ref}(\Pi, \Phi)$ is a choice (partial-order plan) extending Π.]
    if $\mathsf{Ref}(\Pi, \Phi) = \emptyset$ **then**
      [Backtracking process.]
    else
      EVALUATION$(\mathsf{Ref}(\Pi, \Phi))$;
      $\Pi :=$ SELECTION$()$;
return fail; *[Not exists plan.]*

The state-variable representation used in DeLP-MAPOP is based on the latest PDDL (Planning Domain Definition Language) version, PDDL3.1 [15], which was introduced in the context of the 2008 International Planning Competition. Here, we extend the language PDDL3.1 for supporting the specification of defeasible rules and the ambient agent's preferences. Moreover, our language allow us to specify binary variables. A state variable v_i is interpreted in PDDL3.1 as a function that represents a characteristic shared by some of the objects that define the problem. v_i is a tuple that takes the following form $v_i = (vN_i\ p_1 \ldots p_n)$, where vN_i is the unique variable's name and $p_1 \ldots p_n$ are the objects as input parameters of the function. For instance, let *pos-t11* be a variable that indicates the current position of the medical team *t11*; this variable is encoded in PDDL3.1 through the function *(pos t11)*, where '*pos*' is the function name and *t11* is the function parameter. An assignment of a value vl_i to a variable v_i in PDDL3.1 is denoted by (`assign` $\mathtt{v_i}$ $\mathtt{vl_i}$); and the comparison operation is represented by $(= \mathtt{v_i}\ \mathtt{vl_i})$. We also allow to express multi-valued variables for ease of coding, denoted by (`member` $\mathtt{v_i}$ $\mathtt{vl_i}$). For simplicity, we will use the notation <variable,value> in the explanations and use the PDDL3.1 language only to show the encoding of the defeasible rules and planning actions of the planning task. All of these encodings will be shown in a framed box labeled with the caption name Listing.

[1]The open goal Φ is selected as the most costly open goal according to a reachability analysis of the variables.

3.1 Overview of the Application Scenario

This section provides a brief overview of the AmI application upon which the framework DeLP-MAPOP is applied. The purpose is to motivate the interest of this type of applications as well as the utilization of a defeasible planning model to carry out the necessary operations to fulfill the user's need at a specific time.

Nowadays, more and more patients are suffering heart diseases which is the main cause of premature death. The monitoring of people suffering heart failure is currently a challenge for AmI systems. The work in [6] presents a first approach to use an AmI system with centralized planning capabilities for assisting patients suffering diabetics problems. Here, we assume that the patient's home is equipped with appropriate technologies to create the AmI environment. The patient is monitored with a system, in the form of a bracelet, which collects the patient's physical activity and wirelessly transmits it to a device responsible for monitoring patient's heart rate. When a need is detected by this device, e.g. an extremely lower level of a patient's physical activity which may end up in a heart attack, the AmI environment executes DeLP-MAPOP for assisting the patient until the health services arrive to the patient's home.

In this application, we have the following ambient agents: a communication agent in charge of using telecommunication devices such as a cell telephone to call the emergency services; the assistant agent, who is responsible for controlling an automated external defibrillator, an activity tracking device, a position tracking device, etc. to interact with both the environment and the user; and the transport agent, whose main function is to guide the ambulance/helicopter to follow the best path to reach the patient's home. Agents have different capabilities according to their role so they contribute to the overall plan with different actions accordingly. However, we assume that agents' beliefs concern any aspect of the context information and so agents can make assumptions on the current status of the application regarding any type of information. That is, beliefs are not necessarily related to the planning capabilities of the agent, they can refer to any aspect of the AmI environment. The hospitals' preferences are associated with the transport-agent specific preferences, while the patient's preferences are related to the specific preferences of the assistant agent. For space reasons, we omit the specification of the planning task of each ambient agents.

3.2 Plan proposals process

At the PROPOSALS stage, agents generate their refinements $\mathsf{Ref}(\Pi, \Phi)$ to solve an open goal Φ in a partial plan Π, similarly to a plan-space planning process that builds a POP tree, except that each refinement or successor of Π may be now generated by a different agent. Another distinguishing characteristic of the partial order plans generated in DeLP-MAPOP is that they also contain argument steps, as explained in section 2.5, to support action preconditions; this argument structure formed in each partial plan will be later used in the EVALUATION process. The PROPOSALS stage finishes when all agents have made their plan proposals at their turn and these are communicated to the rest of agents. Then, agents update their set of actions with the information appearing in the refinements proposed by the other agents.

Let's suppose an ambient agent Ag_1 who has transport capabili-

ties and knows there are three hospitals in the city {*H1, H2, H3*}. Each hospital disposes of two ambulances from *{a11, a12 . . ., a32}* (one equipped with an Advanced Life Support (ALS) equipment, and the other equipped with a Basic Life Support (BLS) equipment) and one emergency helicopter from *{h1, . . . h3}*. Moreover, Ag_1 knows there are always two emergency medical teams from the set *{t11, t12 . . ., t32}* on call in each hospital: one handles the ALS emergency equipment, and is formed by an ambulance driver, a nurse and a physician; the other handles the BLS equipment and is formed by an ambulance driver and a nursing assistant. Ag_1 also has the defeasible rule specified in Listing 1 and the planning action shown in Listing 2, among others. Note that the new location of the ambulance and the medical-team are generated through the defeasible rule moved-medical-assistance, which is embedded in an argument whose base must be supported by the effects of the action moving-medical-assistance. This allows agents to intervene during the argumentative dialogue in the EVALUATION stage to defeasibly attack the intended effects of the planning action; that is, in case agents have beliefs that make them conclude that the action would not achieve its expected effects.

```
(:def-rule moved-medical-assistance
    :parameters(?a - ambulance
    ?a1 address-hospital ?a2 - address-patient-home
    ?m - medical-team)
    :head (and (assign (at ?a) ?a2)
               (assign (pos ?m) ?a2))
    :body (and (= (moved-amb ?a ?a1) ?a2)
               (= (moved-team ?m ?a1) ?a2)))
```

Listing 1: The body of the defeasible rule matches the effects of the action moving-medical-assistance to deal with the qualification problem.

```
(:action moving-medical-assistance
   :parameters (?a - ambulance
    ?a1 address-hospital ?a2 - address-patient-home
    ?m - medical-team ?t - support-type)
    :effect (and (assign (moved-amb ?a ?a1) ?a2)
                 (assign (moved-team ?m ?a1) ?a2))
    :precondition (and (member (link ?a1) ?a2)
                       (member (type ?t) ?m)
                       (member (contains ?t) ?a)
                       (= (at ?a) ?a1)
                       (= (pos ?m) ?a1)))
```

Listing 2: An action for moving an ambulance from a location to other one.

Let pH be the patient's home. If Ag_1 is asked to solve the open goal $\mathsf{P}(\alpha_G) = \langle$ *at-?a, pH* $\rangle$ ('?a' is an ambulance) generated by the AMS agent to assist the patient, Ag_1 generates at least 6 refinement plans (3 hospitals * 2 ambulances) by using Listings 1 and 2 at the PROPOSALS stage. One of these proposed refinement plan generated is $\Pi_r^{\mathsf{Ag}_1}$, such that $\mathcal{OC}(\Pi_r^{\mathsf{Ag}_1}) = \{\alpha_\Psi \prec \alpha_1; \alpha_1 \prec \mathcal{A}^{\mathsf{Ag}_1}; \mathcal{A}^{\mathsf{Ag}_1} \prec \alpha_G\}$, as shown graphically in Figure 1(b); in this particular example:

- $\mathsf{concl}(\mathcal{A}^{\mathsf{Ag}_1}) = \{\langle$ *pos-t11, pH* $\rangle, \langle$ *at-a11, pH* $\rangle\}$ matches $\mathsf{P}(\alpha_G)$.
- $\mathsf{X}(\alpha_1) = \{\langle$ *moved-amb-a11-H1, pH* $\rangle, \langle$ *moved-team-t11-H1, pH* $\rangle\}$ matches $\mathsf{base}(\mathcal{A}^{\mathsf{Ag}_1})$.

Therefore, argument $\mathcal{A}^{\mathsf{Ag}_1}$ is indirectly deriving the effects of the action α_1. However, unlike non-argumentative MAP systems, in DeLP-MAPOP the open goal $\langle$ *at-?a, pH* $\rangle$ can also be derived by means of an argument that an agent, say Ag_2, puts forward to indicate, that according to its knowledge, ambulance *a31* is already at the patient's home. The base of this argument may be supported with the information provided by an ambulance position tracking device which allows Ag_2 to infer that an ambulance is already located at the patient's home. As the rest of agents do not own this information, they would claim for the inclusion of an action that moves an ambulance to the indicated place. Therefore, unlike classical planning, the argumentation mechanism in DeLP-MAPOP enables supporting an open goal with the context information of an agent without having to necessarily include an action to satisfy such goal, which results in less costly plans. The next stage would show the procedure to guarantee that a goal is satisfactorily warranted by an argument.

3.3 Plan evaluation process

At the EVALUATION stage, agents become engaged in a number of argumentative dialogues aimed at evaluating the guarantee of a successful execution of a plan proposal, i.e the possibility that the actions' intended effects or the derived information, both represented as argument steps in the plan proposal under evaluation, are not achieved as a result of AmI environment changes.

The input of this process is $\mathsf{Ref}(\Pi, \Phi)$, the set of plans proposed by the agents at the previous plan proposal stage. Since ambient agents may have different available context information (represented as a combination of facts and defeasible rules) depending on their information sources, they may not agree on the evaluation of a plan proposal at some point during the dialogue. The EVALUATION stage generates as many argumentative dialogues as argument steps are present in the proposal plan under evaluation. An argumentative dialogue is an exchange of arguments for or against the fulfillment of an argument step, represented as a Plan Argument Dialogue (PAD) tree $\mathcal{T}_{\Pi_r}^{\mathcal{A}}$, where $\Pi_r \in \mathsf{Ref}(\Pi, \Phi)$ is the refinement plan to be evaluated and $\mathcal{A} \in \mathcal{AR}(\Pi_r)$ is the particular argument step to be evaluated. We denote the nodes in a PAD tree as tuples of the form $(\Pi_r, \mathcal{A}, \Gamma)$, where Γ is a set of attacking arguments (whose bases are warranted in the plan Π_r) that will finally determine if argument $\mathcal{A}$ is warranted in plan Π_r. Every node in a PAD tree (except the root) represents a defeater of its parent, and the leaves of the tree correspond to undefeated plans. The set of direct successors nodes of a given node Π_r, is denoted as $\mathsf{succ}(\Pi_r)$. More specifically:

1. The root of the tree is labeled with $(\Pi_r, \mathcal{A}, \emptyset)$.
2. A plan node $(\Pi_r, \mathcal{A}, \{\mathcal{B}\}) \in \mathsf{succ}(\Pi_r)$ represents an attack against the argument $\mathcal{A}$ in plan Π_r through the inclusion of an attacking argument, namely $\mathcal{B}$. Consequently, each node in $\mathsf{succ}(\Pi_r)$ stands for a defeater of the root argument $\mathcal{A}$, i.e. $\mathcal{B}$ is a defeater of $\mathcal{A}$.
3. A plan node $(\Pi_r, \mathcal{A}, \{\mathcal{B}, \mathcal{C}\}) \in \mathsf{succ}(\mathsf{succ}(\Pi_r))$ indicates an attack to the argument child $\mathcal{B}$ of the parent node through the inclusion of a new attacking argument, say $\mathcal{C}$, so this new node is a supporter of the root argument $\mathcal{A}$.

Informally we might see a PAD tree for an argument step $\mathcal{A}$ as generating a dialectical tree [5] for $\mathcal{A}$. But in DeLP-MAPOP the nodes in the PAD tree are contextualized within a plan. Every linear path from the root to a leaf corresponds to one different acceptable argumentation line. Circular argumentation (also known as fallacious argumentation) is avoided by applying both conditions from [5]: no argument can be reintroduced in the same argumentation line and argument concordance must be guaranteed.

Let Ag_2 be an agent that has the defeasible rules detailed in Listing 3, and $\{\langle$*device-measure-the-traffic-H1-pH, high* $\rangle, \langle$ *maps-google-distance-H1-pH, long* $\rangle\} \subseteq \Psi_{\mathsf{Ag}_2}$. When Ag_1 sends the PAD tree $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_1}}$ (containing only the root node) to the rest of agents, Ag_2 puts forward an attacking argument $\mathcal{B}^{\mathsf{Ag}_2} = (\{\langle$ *pos-t11, H1* $\rangle\}, \{\delta_0; \delta_1; \delta_2\})$, inspired by Listing 3, where:

- δ_0 =*(and ⟨ pos-t11, H1 ⟩ ⟨ at-a11, H1 ⟩)* $\prec$ *(and ⟨ moved-amb-a11-H1, pH ⟩ ⟨ moved-team-t11-H1, pH ⟩ ⟨ traffic-jam-between-H1-pH, true ⟩ ⟨ is-far-from-H1-pH, true ⟩).*
- δ_1 =*⟨ traffic-jam-between-H1-pH, true ⟩* $\prec$ *⟨ device-measure-the-traffic-H1-pH, high ⟩.*
- δ_2 =*⟨ is-far-from-H1-pH, true ⟩* $\prec$ *⟨ maps-google-distance-H1-pH, long ⟩.*

which attacks $\mathcal{A}^{\mathsf{Ag}_1}$. Unlike agent Ag_1, agent Ag_2 knows that traffic jam is expected according to a smart device from the AmI system that monitors the traffic density between the hospital *H1* and the patient's home *pH*, and also knows that the distance between them provided by a web mapping service as Google Maps, is rather large. Both informations may be a reason to believe that an ambulance, initially located at the hospital *H1* will not arrive to *pH* in time for assisting the patient. Thus, Ag_2 creates a new node $(\Pi_r^{\mathsf{Ag}_1}, \mathcal{A}^{\mathsf{Ag}_1}, \{\mathcal{B}^{\mathsf{Ag}_2}\}) \in \mathsf{succ}(\Pi_r^{\mathsf{Ag}_1})$ among others, and sends it to rest of agents.

```
(:def-rule moved-medical-assistance-denied
    :parameters(?a - ambulance
    ?a1 address-hospital ?a2 - address-patient-home
    ?m - medical-team)
    :head (and (assign (at ?a) ?a1)
               (assign (pos ?m) ?a1))
    :body (and (= (moved-amb ?a ?a1) ?a2)
               (= (moved-team ?m ?a1) ?a2)
               (= (traffic-jam-between ?a1 ?a2) true)
               (= (is-far-from ?a1 ?a2) true)))
(:def-rule traffic-jam
    :parameters(?a1 address-hospital
    ?a2 - address-patient-home)
    :head (assign (traffic-jam-between ?a1 ?a2) true)
    :body (= (device-measure-the-traffic ?a1 ?a2) high))
(:def-rule distance
    :parameters(?a1 address-hospital
    ?a2 - address-patient-home)
    :head (assign (is-far-from ?a1 ?a2) true)
    :body (= (maps-google-distance ?a1 ?a2) long))
```

Listing 3: Defeasible rules for representing situations in which the ambulance may not arrive on time.

In the next round of the dialogue, $(\Pi_r^{\mathsf{Ag}_1}, \mathcal{A}^{\mathsf{Ag}_1}, \{\mathcal{B}^{\mathsf{Ag}_2}\})$ is received by the ambient agent Ag_3 who discovers a new attacking argument $\mathcal{C}^{\mathsf{Ag}_3}$ that defeats $\mathcal{B}^{\mathsf{Ag}_2}$, which is based on Listing 4.

```
(:def-rule carpool-lane
    :parameters(?a1 address-hospital
    ?a2 - address-patient-home)
    :head (assign (traffic-jam-between ?a1 ?a2) false)
    :body (= (carpool-lane-between ?a1 ?a2) true))
```

Listing 4: The defeasible rule used for representing a carpool lane which may prevent an ambulance from being stuck by a traffic congestion situation.

Assuming that *⟨ carpool-lane-between-H1-pH, true ⟩* $\in \Psi_{\mathsf{Ag}_3}$, then $\mathcal{C}^{\mathsf{Ag}_3}$ = ({*⟨ traffic-jam-between-H1-pH, false ⟩*},{*⟨ traffic-jam-between-H1-pH, false ⟩* $\prec$ *⟨ carpool-lane-between-H1-pH, true ⟩*}). That is, Ag_3 knows that there is a carpool lane (as an express lane) between *H1* and *pH*, which is a reason to believe that the ambulance *a11* can skip the traffic congestion on the way to reach the patient's home. Ag_3 creates a new plan $(\Pi_r^{\mathsf{Ag}_1}, \mathcal{A}^{\mathsf{Ag}_1}, \{\mathcal{B}^{\mathsf{Ag}_2}, \mathcal{C}^{\mathsf{Ag}_3}\})$ extending $(\Pi_r^{\mathsf{Ag}_1}, \mathcal{A}^{\mathsf{Ag}_1}, \{\mathcal{B}^{\mathsf{Ag}_2}\})$ with $\mathcal{C}^{\mathsf{Ag}_3}$, and sends it to the rest of agents. The evaluation dialogue for $\mathcal{T}_{\Pi_r}^{\mathcal{A}^{\mathsf{Ag}_1}}$ continues until all defeaters are put forward in a round.

In order to check whether the argument of the root node is defeated or undefeated, the following procedure on the PAD tree is applied: label with a *U* (for *undefeated*) each terminal plan in the tree (i.e. each plan with no defeaters at all). Then, in a bottom-up fashion, we label a node with: *U* if each of its successors is labeled with a *D*; and *D* (for defeated) otherwise.

A plan in $\mathsf{Ref}(\Pi, \Phi)$ is labeled as an **undefeated refinement plan** if all the root plans of its PAD trees are labeled as undefeated. Otherwise the plan is provisionally labeled as a **defeated refinement plan** in the POP tree. Undefeated plans are obviously preferred over defeated plans as they represent a plan with no expectation failures according to the ambient agents. Nevertheless, defeated plans are maintained in the POP tree as their arguments may become later undefeated as the problem evolves and information changes. Finally, each ambient agent updates its initial facts and defeasible rules with the facts and defeasible rules from the exchanged arguments' bases.

3.4 Plan selection process

At the SELECTION stage, the aim is to select the next plan Π to be refined and continue with the plan-space planning process of the PROPOSALS stage, unless Π is already a solution in which case the DeLP-MAPOP protocol stops.

For selecting a plan, agents apply three criteria in order of priority over the set of evaluated plans from the previous stage. The objective is to select a plan considering a compromise between the desire to minimize the computational overhead and that of maximizing the quality of the solution plan. The three criteria are: first, the system applies a warranty procedure to discard the plans evaluated as defeated in the evaluation stage. Second, a heuristic function is applied over the undefeated plans resulting from the above filtering. We use two of the most popular heuristics in planning: SUM and MAX heuristics [16]. The SUM heuristic estimates the cost of a plan as the sum of the cost of the pending open goals in the plan whereas the MAX heuristic returns the value of the most costly open goal as heuristic estimation. Plans whose heuristic estimation is below a certain threshold are discarded from consideration. Finally, the last filtering over the remaining plans considers the preference functions. We have implemented two intersection techniques aimed at selecting the most preferable plan by the ambient agents according to their preferences. The first mechanism selects the plan whose actions are all among the preferences of every agent with a degree of preference above a certain threshold. If the application of this method returns an empty list then we compute the number of preferred actions in each plan and we select the plan with the largest proportion of preferred actions by the ambient agents.

4. EXPERIMENTAL EVALUATION

The purpose of this section is to test the overall performance, scalability and quality of DeLP-MAPOP versus a MAP system with no argumentation (MAPOP) which has also been implemented in the same agent platform, and discuss the benefits and limitations of each system. We carried out several experiments considering three different levels of difficulty of the planning problem: small (composed by 8 grounded actions and 50 grounded defeasible rules), medium (composed by 16 grounded actions and 100 grounded defeasible rules) and large (composed by 24 grounded actions and 150 grounded defeasible rules). We used teams of agents of different size ranging from 1 (single-agent) to 5. We performed several tests varying the number of agents of each type in the AmI environment, namely transportation, communication and assistant agents, and we took the median values over 20 repetitions for each set of experiments with 'n' agents, regardless the type of agent. We used the MAX heuristic and the Intersection function.

DeLP-MAPOP and MAPOP are implemented on Magentix2[2], a multi-agent platform based on Apache Qpid[3], an open-source implementation of Advanced Message Queuing Protocol for communication.

With regard to scalability and performance, Figures 2(a), 2(b) and 2(c) show the average time spent on each stage of the DeLP-MAPOP protocol, while Figure 2(d) shows the average total time to find a solution plan, including parsing the problem file and grounding the planning actions and defeasible rules. The horizontal axis (the same for the rest of the figures) depicts the size of the team of ambient agents, while the vertical axis displays the time in milliseconds. As expected, the average time spent in DeLP-MAPOP is always greater than the time spent in MAPOP due to the following reasons: i) in the PROPOSALS stage, the ambient agents from DeLP-MAPOP do not only have to reason about which actions would achieve the selected open goal, but also need to reason about which arguments would support it; ii) the EVALUATION stage is not considered in MAPOP; and iii) the SELECTION stage is replaced in MAPOP by a single heuristic function. It is also noticeable that the more agents in a team, the more exchanged messages between them, causing each stage to take longer in DeLP-MAPOP. Figure 2(e) illustrates precisely that, as the number of agents increases, the number of exchanged messages is larger; Figure 2(f) shows that as the size of the team increases, the number of dialogue rounds is lower because in this case more attacking arguments tend to appear in a single round, thus decreasing the number of rounds.

Figure 3 shows the evaluation of the quality of the obtained solution plans. Figure 3(g) shows that the average number of action steps in solution plans of DeLP-MAPOP is lower or equal than the average number in solution plans of MAPOP. The reason is that in MAPOP, an open goal that is not a threat can only be achieved through an action step, while in DeLP-MAPOP the open goal can also be supported by an argument step whose base is already guaranteed in the plan. In these cases, the cost of the DeLP-MAPOP plans is smaller because fewer actions are required to support the open goals, meaning that the agents' beliefs support the fulfillment of a goal without explicitly including an additional action in the plan. The fact that argument steps are not used in MAPOP is precisely shown in Figure 3(h). On the other hand, we can see in Figure 3(i) a comparison of the quality of plans generated with a single-agent team versus plans generated by teams with more than one agent. Obviously, in the first case, plans are sequential while DeLP-MAPOP returns plans with parallel actions that can be simultaneously executed by different ambient agents.

We also carried out one more experiment: which action steps in MAPOP solution plans are actually discarded during an argumentative dialogue in DeLP-MAPOP plans. This latter aspect is also a very relevant issue as we wanted to compare the plans returned by both systems and see how many plans, and actions correspondingly, of MAPOP were actually discarded by the agents in DeLP-MAPOP during the argumentative dialogues. The results of this experiment are shown in Figure 3(j). As can be seen, according to the knowledge of the ambient agents, 0% of the solution plans generated by DeLP-MAPOP comprise failing actions, i.e. actions whose intended effects were acknowledged to fail at the EVALUATION stage. Obviously, as long as agents acquire more information from the context, argumentative dialogues will fit reality better and, therefore, the guarantee of a successful solution plan (a plan with no expected failures) would also be greater. Furthermore, this experiment allowed us to check the correctness of the argumentative dialogues at the EVALUATION stage. However, in the case of MAPOP, up to 50% of the plans had actions that were discarded by the ambient agents in DeLP-MAPOP, that is, actions that agents acknowledged that would not be successfully executed.

Examining the influence of preferences in DeLP-MAPOP, Figure 3(k) shows that the average satisfaction of each team with the solution plans decreases as the size of the team increases. We calculated the satisfaction of an individual agent on a solution plan by averaging its preferences in the action steps of the plan, while the team satisfaction is calculated as the average of the individual satisfactions. Figure 3(l) shows that the difference of satisfaction between agents tends to increase as the size of the teams also increases. It is desirable that the difference is as small as possible for that all agents are equally satisfied.

[2] http://www.gti-ia.upv.es/sma/tools/magentix2/index.php

[3] http://qpid.apache.org/

5. CONCLUSIONS AND FUTURE WORK

This paper presents the specification, implementation and an exhaustive experimentation of DeLP-MAPOP, an argumentation-based defeasible planning framework, on AmI applications. Our most relevant contribution is a fully implemented MAP framework that has been extensively tested in AmI environments. DeLP-MAPOP realizes three independent but cooperative processes to propose, criticize, defend and select alternative plan proposals. The results show two advantages of DeLP-MAPOP over a MAP process with no argumentation: (i) since each plan step of a plan proposal is collaboratively argued, DeLP-MAPOP returns plans whose actions are not likely to fail at execution time according to the information and beliefs of the ambient agents; and (ii) since open goals can also be supported by argument steps whose base is warranted with the facts of the plan, the context information and defeasible reasoning of agents provide a means to satisfy goals of the problem without an explicit inclusion of a planning action; this avoids considering unnecessary action steps thus reducing the total cost of the plan.

As future work, we intend to test the effectiveness and feasibility of DeLP-MAPOP in a hospital pilot program, as well as an extension to temporal defeasible argumentation for MAP [17]. Currently, we are working on the development of a more elaborated heuristic function that (i) analyzes the transitions between the values a state variable can take, and (ii) considers the experiences from the plan evaluation process (case-based argumentation) to predict the potential number of attacks that a plan can receive. We are also interested in studying the influence of the trust on the sources (devices) used by the ambient agents to acquire the context information as well as how a trust level determines the conflict resolution between attacking arguments. Finally, a comparison with other MAP approaches will be considered.

Acknowledgements

This work is mainly supported by the Spanish Ministry of Science and Education under the FPU Grant reference AP2009-1896 awarded to Sergio Pajares Ferrando, and projects, TIN2011-27652-C03-01, Consolider Ingenio 2010 CSD2007-00022, and PROMETEO/2008/051.

6. REFERENCES

[1] E. Aarts. Ambient intelligence. *Adaptive Hypermedia and Adaptive Web-Based Systems*, Springer, pp. 548–568, 2004.

[2] A. Bikakis and G. Antoniou. Distributed defeasible contextual reasoning in ambient computing. *Ambient Intelligence*, Springer, pp. 308–325, 2008.

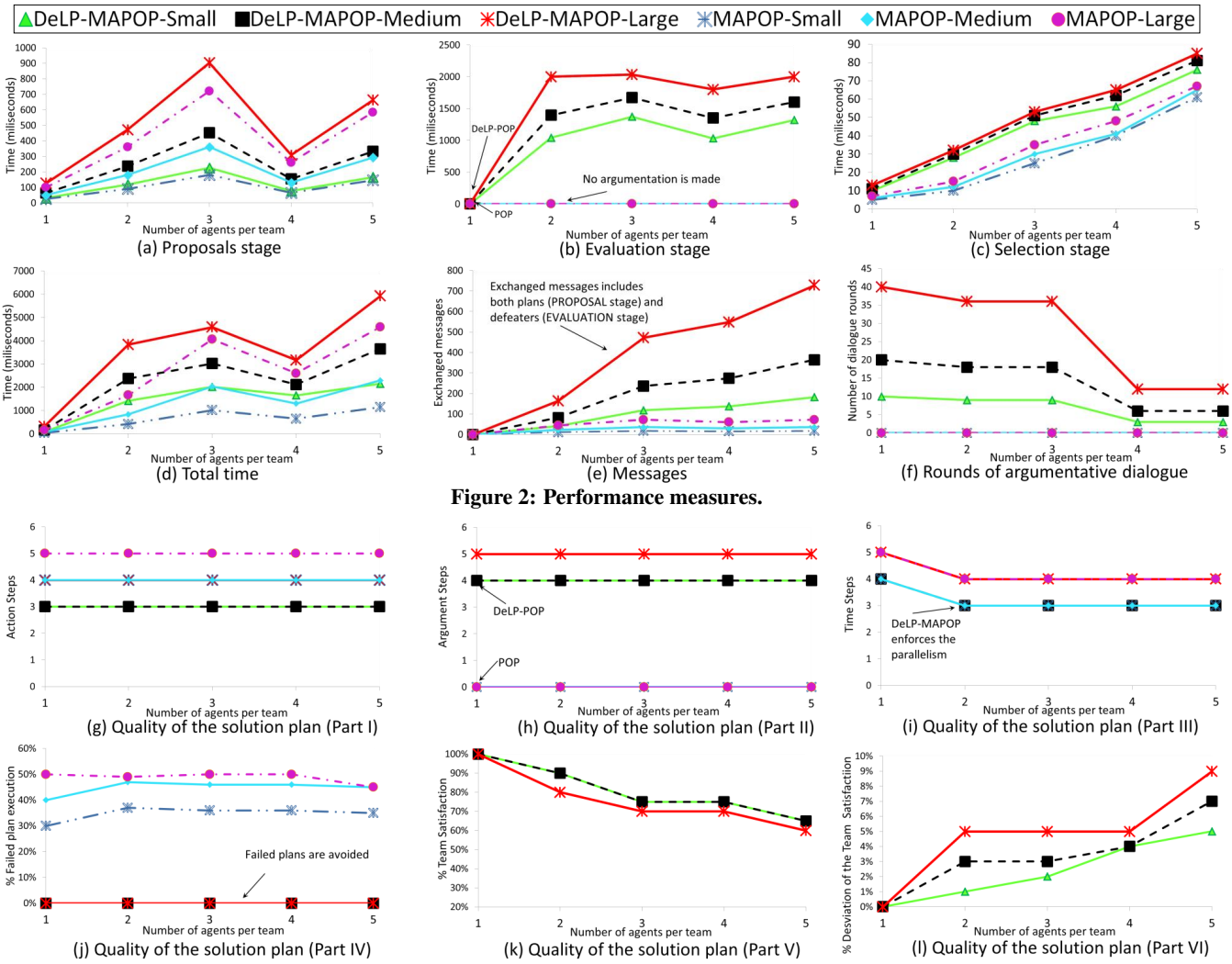


Figure 2: Performance measures.

Figure 3: Quality measures.

[3] H. Prakken and G. Vreeswijk. Logics for defeasible argumentation. *Handbook of philosophical logic*, pp. 4:218–319, 2002.

[4] A. Bikakis and G. Antoniou. Defeasible contextual reasoning with arguments in ambient intelligence. *IEEE Transactions on Knowledge and Data Engineering*, pp. 1492–1506, 2010.

[5] A. García and G. Simari. Defeasible logic programming: An argumentative approach. *Theory and Practice of Logic Programming*, pp. 4:95–138, 2004.

[6] F. Amigoni, N. Gatti, C. Pinciroli and M. Roveri. What planner for ambient intelligence applications?. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, pp. 35(1):7–21, 2005.

[7] J.S. Penberthy and D.S. Weld. UCPOP: A Sound, Complete, Partial Order Planner for ADL. In Proc. of *KR*, pp. 103–114, 1992.

[8] D.E. Smith, J. Frank and A.K. Jónsson. Bridging the gap between planning and scheduling. *The Knowledge Engineering Review*, pp. 15(1):47–83, 2000.

[9] A. Jonsson, P. Morris, N. Muscettola, K. Rajan, and B. Smith. *Planning in interplanetary space: Theory and practice*. In Proc. of *ICAPS*, pp. 177–186, 2000.

[10] D. García, A. García, and G. Simari. Defeasible reasoning and partial order planning. In Proc. of the *International Conference on Foundations of information and knowledge systems, FoIKS 2008*, LNCS 4932, pp. 311–328, 2008.

[11] S. Pajares and E. Onaindía. Defeasible Planning through Multi-Agent Argumentation. *Modelling Machine Emotions For Realizing Intelligence, Smart Innovation Systems and Technologies Series*, pp. 13:311–342, 2011.

[12] P. Pardo, S. Pajares, E. Onaindía, L. Godo and P. Dellunde. Multiagent Argumentation for Cooperative Planning in DeLP-POP. In Proc. of *AAMAS*, pp. 971–978, 2011.

[13] S. Pajares, E. Onaindía and A. Torreño. An Architecture for Defeasible-Reasoning-Based Cooperative Distributed Planning. In Proc. of *CoopIS in conjunction with OTM*, pp. 200–217, 2011.

[14] M.L. Ginsberg and D.E. Smith. Reasoning about action II: The qualification problem. *Artificial Intelligence*, pp. 3:311–342, 1998.

[15] D.L. Kovacs. Complete BNF description of PDDL3.1. *Technical report*, 2011.

[16] X.L. Nguyen, and S. Kambhampati. Reviving partial order planning. In Proc. of *IJCAI*, pp. 17:459–466, 2001.

[17] S. Pajares and E. Onaindía. Temporal Defeasible Argumentation in Multi-Agent Planning. In Proc. of *IJCAI* , pp 2834–2835, 2011.

Personalizing Communication about Trust

Andrew Koster
IIIA-CSIC,
Universitat Autònoma de Barcelona
Spain
andrew@iiia.csic.es

Jordi Sabater-Mir
IIIA-CSIC
Spain
jsabater@iiia.csic.es

Marco Schorlemmer
IIIA-CSIC,
Universitat Autònoma de Barcelona
Spain
marco@iiia.csic.es

ABSTRACT

Agents in open multi-agent systems must deal with the difficult problem of selecting interaction partners in the face of uncertainty about their behaviour. This is especially problematic if they have to interact with an agent they have not interacted with before. In this case they can turn to their peers for information about this potential partner. However, in scenarios where agents may be evaluated according to many different criteria for many different purposes, their peers' evaluations may be mismatched with regards to their own expectations. In this paper we present a novel method, using an argumentation framework, that allows agents to discuss and adapt their trust model. This allows agents to provide, and receive, personalized trust evaluations, better suited to the agent in need, as is shown in a prototypical experiment.

Categories and Subject Descriptors

Computing Methodologies [**Artificial Intelligence**]: Distributed Artificial Intelligence

General Terms

Algorithms, Experimentation

Keywords

Trust, reliability and reputation; Argumentation

1. INTRODUCTION

In any society of intentional agents, trust is an essential tool for selecting interaction partners. Trust is a personal and subjective evaluation of a target for the fulfillment of a specific goal. However, to choose a partner based on trust, an agent needs information about it. In any environment where it does not have direct experiences with interaction partners, it turns to external sources to aid in making this selection. Reputation is one source of such information. Reputation is what a group of individuals say about an agent, regarding its behaviour. Computational models of reputation usually obtain this by aggregating reported evaluations from a large number of individual agents. If the aggregation is performed properly, this can be an effective estimate of an agent's trustworthiness, precisely because it is an aggregation of a large number of reported evaluations.

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

1.1 Reputation and Recommendation

The drawback of reputation that we focus on in this article is that, if an agent has specific requirements, reputation calculated as an aggregation of opinions can be an inadequate measure to decide whether or not a target is able to meet these requirements. This is especially true in any domain where there are many different requirements an agent may have for any specific task. A good reputation for fulfilling that task does not guarantee that the agent complies with a single agent's specific requirements, just that it is generally able to comply with agents' needs.

For tasks such as buying an item in an online marketplace these measures are generally good enough: the range of requirements between agents does not vary much, and a good reputation generally means an agent is good over the entire range of requirements. However, the same cannot be said when the range of requirements increases. When a choice depends on many different criteria, collaborative filtering mechanisms may offer a solution. These mechanisms provide personalized recommendations by relying on large numbers of agents and matching the requesting agent's profile to that of agents who have provided evaluations [17]. These recommendations are more tailored to an individual's needs than reputation, because the system only uses evaluations from agents similar to the requesting agent. Such mechanisms, however, have their own drawbacks. The first is that they require a large number of agents providing not only evaluations, but also a profile (representing the context in which they made the evaluation and the goal they were trying to achieve).

As multi-agent systems mature and gain in popularity, the domains in which they may be applied increase, and not all of these have a large number of agents providing profiles and evaluations. An approach that does not rely on a large network is to use an agent's own network of friends, in order to give personalized recommendations. This is the approach we adopt in this paper.

1.2 Personalized Trust Recommendations

We assume an agent has a computational trust model to aid with partner selection. Ultimately, computational trust models and recommender systems have a very similar aim. Both computational trust models and recommender systems aim to provide accurate evaluations of other agents in order for the agent to select a good partner, in the given context, to achieve its goal.

The fundamental difference is what agent the calculation centers on. Recommender systems aim to tailor a personalized recommendation to the requesting agent. Trust mod-

els, on the other hand, take only the calculating agent's goals and beliefs into account when calculating a trust evaluation. Upon receiving a trust evaluation this needs to be taken into account: it is the sender's subjective and personal evaluation and may, therefore, not be useful to the requesting agent.

In this paper we propose a novel approach to communicate about trust by learning a lesson from recommender systems: we propose to personalize trust evaluations to the requesting agent's requirements. Our proposal is two-pronged. Firstly, agents requesting information may communicate the goal, and corresponding criteria, for which they need a partner to perform a task. The supplying agent can then use this goal and criteria to tailor a trust evaluation to it. Secondly, agents may attempt to persuade each other that their beliefs about the environment, and the corresponding criteria in their trust model, are incorrect. By combining these techniques agents can communicate about personalized trust, allowing them to better estimate a potential partner's performance for fulfilling a specific goal, given the agent's beliefs about the environment.

2. RELATED WORK

Using other agents' trust evaluations directly is not a new idea. A long-standing problem with such communication has been lying or colluding agents. In such cases an agent intentionally communicates a wrong trust evaluation. A solution is to filter out communication from such agents [19]. The underlying assumption is that lying is the only reason other agents' trust evaluations can be mismatched and thus, by detecting agents whose evaluations do not match the own evaluations, the problem can be solved. However, in an environment in which agents may use many different criteria to evaluate each other, these methods will mark many agents as liars who simply have a different opinion.

Koster et al. address this problem by translating others' trust evaluations into the own frame of reference using a machine learning algorithm [8]. However, as with any such algorithm, this requires a large amount of data. In this case the data consists of targets that both agents have evaluated, and can thus be compared. This assumes both agents have already interacted with many agents in the system. Thus it does not work for agents who are new to the environment, or environments with few agents.

Pinyol et al. [13] address the problem of communication in a manner that does not require a large amount of data, by using argumentation about trust. This allows agents to exchange information about their trust model, and thus each agent can decide whether or not to accept a communicated evaluation. While this does not assume agents are lying if their trust evaluations do not match, it suffers a similar drawback to the methods for detecting lies: it will discard a large amount of information if there are many different criteria on which to base an evaluation, because it may only accept or reject a communicated evaluation. We demonstrate this drawback empirically in Section 6. There are other approaches that combine trust and argumentation [10, 6], but these focus on different aspects of the area.

3. ADAPTRUST

Our method for enabling personalized communication about trust is based on three capabilities an agent must have:

1. An agent must be able to adapt its trust model in order to personalize its evaluations to the other agent's needs.

2. An agent must be capable of communicating its criteria for evaluating trust, as well as the underlying beliefs and goals leading to these criteria.

3. An agent must be willing and able to change its trust model, if it is persuaded that its beliefs about the environment, and thus the criteria for calculating trust are wrong.

We assume that agents are willing to adapt their model if they are convinced it is inaccurate. We use AdapTrust to enable this, and, additionally so agents can adapt their trust model to another agent's needs. AdapTrust is an extension of the BDI framework for intelligent agents [16]. As the name implies, AdapTrust allows a trust model to be adapted, according to an agent's goals and beliefs. We present the method in full detail in [9] and summarize it here.

Computational trust models are, fundamentally, methods of aggregation: they combine and merge data from several different sources into a single value, the trustworthiness of a target. As argued in the introduction, the evaluation of a target is dependent on the *beliefs* the evaluator has about the world, as well as the *goal* it is trying to achieve. Luckily most computational trust models come equipped with a way of implementing this dependency: they have parameters that can be used to adjust the behaviour of the trust model. The aim of AdapTrust is not to present another trust model, but to incorporate existing trust models into an intelligent agent. This can be used to deal with the multifaceted aspects of trust or, as we show in this article, adapt the trust model to improve communication about trust.

In any computational trust model, there are parameters that represent criteria for evaluating trustworthiness. For instance, many trust models use a parameter to give less importance to old information than new. This is useful if old information can become outdated and thus new information is more accurate than old. However, in a largely static environment this is not the case. The value of this parameter should be adjusted to the dynamicity of the environment. In general, the parameters of the trust model should be influenced by an agent's changing criteria for evaluating trustworthiness in a changing environment.

3.1 Priority System

The parameters of a trust model describe the importance of the different criteria for evaluating trustworthiness. However, it is more useful to consider this the other way round: the relative importance between the different criteria define a set of parameters for the trust model. These criteria are directly under an intelligent agent's control, and thus an agent is able to adapt its trust model. AdapTrust describes the specific techniques necessary to do this. The first of these is $\mathcal{L}_{PL}$, a language to describe the relative importance of any two criteria that influence a parameter of the trust model. We chose a subset of first-order logic with a family of predicates to define this importance relation, also called a priority ordering. For each parameter p of the trust model, the binary predicates $\succ_p$ and $=_p$ are defined with the expected properties of strict ordering and equality, respectively. The terms of the language are a set of elements representing the criteria that influence how the trust model should work. A Priority System is defined as a satisfiable theory in this language. For instance, consider an eCommerce environment. If an agent uses a weight w to calculate its evaluation of a sale and it finds the price of an item to be more important than its delivery time, it can have the priority $price \succ_w delivery\_time$ in its Priority System.

3.2 Priority Rules

The second technique of AdapTrust is to create the link between, on the one hand, an agent's beliefs and goals and, on the other hand, the priority between the different criteria for evaluating trust. This link makes explicit the adaptive process: a change in an agent's beliefs or goals effects a change in the priorities over the criteria, which changes the parameters of the trust model. The connection between the beliefs or goals and the priorities is made through what we call *priority rules*. The priority rules are specified using another first-order language, $\mathcal{L}_{Rules}$, with predicates $\rightsquigarrow_{Belief}$ and $\rightsquigarrow_{Goal}$ specifying how a set of beliefs, or a goal, respectively, leads to a specific priority relation between two criteria. By using these rules, we see that when the belief base changes the priorities can change. Additionally this is how the multifaceted aspect of trust is emphasized: the goal the agent is trying to achieve influences the priority system and thus the trust model. For instance, in the eCommerce example above, our agent might need to buy a bicycle urgently. It then has the goal $buy\_urgent(bicycle)$. For this goal, delivery time is more important than the price, so it has the priority rule $buy\_urgent(bicycle) \rightsquigarrow_{Goal} (delivery\_time \succ_w price)$ and therewith *adapts* its trust model to the requirements of the goal.

We do not go into detail on how these priority rules come to be. They can be programmed by a designer, or generated dynamically by a machine learning algorithm. However, what we are interested in here, is that they can also be incorporated through communication with another agent. We will return to this in Section 5, but first we describe the basic argumentation framework that we extend to allow for this communication. For a full description of the AdapTrust mechanism we refer an interested reader to [9].

4. PINYOL'S ARGUMENTATION METHOD

In the previous section we addressed two of the three requirements for agents to provide personal communications about trust. The last is that they are able to communicate about their criteria for evaluating trust. These criteria are given by an agent's beliefs and goals. What we need is thus a communication language that allows agents to talk about trust evaluations, the beliefs and goals these depend on, and the causal relationship between the two. We present this in Section 5, however Pinyol proposed a partial solution to this problem: an information-seeking dialogue for communication about trust [14]. The main aim of this framework is to allow the receiver of a communicated evaluation to decide whether or not to accept it. The framework creates an argument abstracting away from the computational process of the trust model, thereby allowing agents to discover what the original sources for evaluating a trust evaluation are. However, when asked *why* an aggregation of sources resulted in a specific evaluation, the model can only repeat itself as this is modeled as a ground element of the argumentation language. Our proposal extends the framework and allows agents to answer such questions, but first we summarize Pinyol's argumentation framework.

4.1 Trust as an inferential process

Pinyol starts by modeling the trust model as an inference relation between sentences in $\mathcal{L}_{Rep}$, a first-order language about trust and reputation [14]. This language is defined by a taxonomy of terms used for describing the process of computing trust. A trust model is considered as a computational process: given a finite set of inputs, such as beliefs about direct experiences or reputation, it calculates a trust evaluation for a target. The semantics of a computational process can be given by the application of a set of inference rules [7]. We define this as follows:

DEFINITION 1 (SEMANTICS OF A TRUST MODEL). *We say that a set of inference rules $\mathcal{I}$ is a specification of a trust model if, given input Δ and the resulting trust computation δ, we have that $\Delta \vdash \delta$, i.e., there exists a finite number of applications of inference rules $\iota \in \mathcal{I}$ by which we may infer δ from Δ.*

The inference rules themselves depend on the specifics of the computational process and thus the actual trust model being used, but for any computational trust model, such an inference relation exists. For instance, a trust model might have a rule:

$$\frac{img(T, X), rep(T, Y)}{trust(T, \frac{X+Y}{2})}$$

With img, rep and $trust$ predicate symbols in $\mathcal{L}_{Rep}$ and T, X and Y variables. For a specific target Jim, an agent knows $\{img(Jim, 3),\ rep(Jim, 5)\}$. It can thus infer $trust(Jim, 4)$ using the rule above. For a full example of representing a trust model in inference rules, we refer to [12].

4.2 Arguing about trust

Arguments are sentences in the $\mathcal{L}_{Arg}$ language. This language is defined over another language $\mathcal{L}_{KR}$, that represents object-level knowledge . In Pinyol's framework $\mathcal{L}_{KR} = \mathcal{L}_{Rep}$, but in Section 5 we will supplement this language in order to extend the argumentation. A sentence in $\mathcal{L}_{Arg}$ is a formula $(\Phi : \alpha)$ with $\alpha \in \mathcal{L}_{KR}$ and $\Phi \subseteq \mathcal{L}_{KR}$. This definition is based on the framework for defeasible reasoning through argumentation, given by Chesñevar and Simari [4]. This framework of argumentation provides a clear manner for constructing arguments from an underlying language, rather than just providing a way for resolving what set of arguments fulfill certain criteria, which is the usual role of an argumentation framework [5, 2]. An alternative could be to model the trust model using a bipolar argumentation framework [1], however we choose to follow Pinyol's approach, which we explain here. Intuitively Φ is the defeasible knowledge required to deduce α. Defeasible knowledge is the knowledge that is rationally compelling, but not deductively valid. The meaning here, is that using the defeasible knowledge Φ and a number of deduction rules, we can deduce α. The defeasible knowledge is introduced in a set of elementary argumentative formulas. These are called *basic declarative units*.

DEFINITION 2 (BASIC DECLARATIVE UNITS). *A basic declarative unit (bdu) is a formula $(\{\alpha\} : \alpha) \in \mathcal{L}_{Arg}$. A finite set of bdus is an argumentative theory.*

Arguments are constructed using an argumentative theory Γ and the inference relation $\vdash_{Arg}$, characterized by the deduction rules *Intro-BDU, Intro-AND* and *Elim-IMP*.

DEFINITION 3 (DEDUCTION RULES OF $\mathcal{L}_{Arg}$).

$$\textit{Intro-BDU:}\frac{}{(\{\alpha\} : \alpha)}$$

$$\textit{Intro-AND:}\frac{(\Phi_1 : \alpha_1), \ldots, (\Phi_n : \alpha_n)}{(\bigcup_{i=1}^{n} \Phi_i : \alpha_1 \wedge \cdots \wedge \alpha_n)}$$

$$\textit{Elim-IMP:}\frac{(\Phi_1 : \alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta), (\Phi_2 : \alpha_1 \wedge \cdots \wedge \alpha_n)}{(\Phi_1 \cup \Phi_2 : \beta)}$$

An argument $(\Phi : \alpha)$ is valid on the basis of argumentative theory Γ iff $\Gamma \vdash_{Arg} (\Phi : \alpha)$. Because the deduction rules, and thus $\vdash_{Arg}$, are the same for all agents, they can all agree on the validity of such a deduction, however each agent builds its own argumentative theory, using its own trust model. Let $\mathcal{I}$ be the set of inference rules that specify an agent's trust model. Its bdus are generated from a set of $\mathcal{L}_{Rep}$ sentences Δ as follows:

- For any ground element α in Δ, there is a corresponding bdu $(\{\alpha\} : \alpha)$ in $\mathcal{L}_{Arg}$.
- For all $\alpha_1, \ldots, \alpha_n$ such that $\Delta \vdash \alpha_k$ for all $k \in [1, n]$, if there exists an applicaton of an inference rule $\iota \in \mathcal{I}$, such that $\frac{\alpha_1,\ldots,\alpha_n}{\beta}$, then there is a bdu $(\{\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta\} : \alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta)$, i.e., there is a bdu for every instantiated inference rule for the model specified by $\mathcal{I}$.

$\mathcal{L}_{Arg}$ is a non-monotonic logic and implication is defined in a similar manner to implication in logic programming. For details on the semantics, we refer to Chesñevar and Simari's work [4]. Continuing the example from above, our agent might have bdus:
$(\{img(Jim, 3)\} : \ img(Jim, 3))$,
$(\{rep(Jim, 5)\} : \ rep(Jim, 5))$ and
$(\{img(Jim, 3) \ \wedge \ rep(Jim, 5) \rightarrow trust(Jim, 4)\} :$
$img(Jim, 3) \ \wedge \ rep(Jim, 5) \rightarrow trust(Jim, 4))$.
These bdus constitute an argumentative theory, from which $(\Phi : trust(Jim, 4))$ can be inferred, with Φ the union of the defeasible knowledge of the argumentative theory. Similarly, working backwards, an agent can build a valid argument supporting a trust evaluation it believes. Moreover, it can communicate this argument. This forms the first part of the information-seeking dialogue we need, in order to enable personalized trust communications. The problem, however, is that the trust model's functioning is introduced into the argumentation language in the form of bdus (see above). This means agents cannot explain why their trust model performs a specific calculation, because it is treated as defeasible knowledge. In the next section we present our extension to this framework, that allows agents to explain the reasons for their trust model's functioning.

5. PERSONALIZED TRUST

In this section we first present the extension of the argumentation framework, allowing agents to fully express the importance of criteria in their trust model. Subsequently we provide a dialogue protocol, allowing two agents to compare their trust evaluations, in order to discover where their trust models diverge. We provide a range of options to allow for the adaptation of their trust models, so that one agent can compute a trust evaluation tailored to the personal requirements of the other agent.

5.1 Extending the Argumentation Language

Pinyol's argumentation framework does not allow us to completely address the question of what criteria play a role in computing a trust evaluation, let alone connect these to underlying beliefs and goals. AdapTrust can answer this, but does not provide a language in which to do so. We propose to extend the argumentation framework presented in Section 4 with concepts from AdapTrust. In AdapTrust the *reason* an agent performs this computation and not some other one, is twofold: firstly the trust model follows an algorithmic method for aggregating the input. Secondly, the agent's beliefs and goals fix the parameters of this algorithm.

We do not propose to explain the algorithmic processes in the trust model, but the criteria, given by beliefs and goals, that define the trust model's parameters can be incorporated into the argumentative theory. For this, we need to represent the dependency of the trust model on the beliefs and goal of an agent in $\mathcal{L}_{Arg}$. In $\mathcal{L}_{Rep}$, the inference rules $\mathcal{I}$ specify a trust model algorithm. However, in AdapTrust this algorithm has parameters that depend on the agent's beliefs and goal. The inference rules should reflect this. Let $\Delta \subseteq \mathcal{L}_{Rep}$ and $\delta \in \mathcal{L}_{Rep}$, such that $\Delta \vdash \delta$. From Definition 1 we know there is a proof applying a finite number of inference rules $\iota \in \mathcal{I}$ for deducing δ from Δ. However, this deduction in AdapTrust depends on a set of the parameters, which we denote $Params$. Therefore, the inference rules must also depend on these parameters. For each $\iota \in \mathcal{I}$, we have $Params_\iota \subseteq Params$, the (possibly empty) subset of parameters corresponding to the inference rule and the set of parameters corresponding to a proof $\Delta \vdash \delta$ is simply the union of all parameters of the inference rules used in the deduction. Let the beliefs Ψ and goal γ determine the values for all these parameters. We denote this as $\Delta \vdash^{\Psi,\gamma} \delta$, which states that the trust model infers δ from Δ, given beliefs Ψ and goal γ. Similarly we have $\iota^{\Psi,\gamma} \in \mathcal{I}^{\Psi,\gamma}$ to denote a specific instantiation of the parameters $Params_\iota$ using beliefs Ψ and goal γ.

This allows us to redefine the set of bdus and thus the argumentative theory in such a way that the argumentation supporting a trust evaluation can be followed all the way down to the agent's beliefs and goal. $\mathcal{L}_{KR}$ must thus also be extended to encompass the various languages in AdapTrust, namely $\mathcal{L}_{KR} = \mathcal{L}_{Rep} \cup \mathcal{L}_{PL} \cup \mathcal{L}_{Rules} \cup \mathcal{L}_{Bel} \cup \mathcal{L}_{Goal}$, where $\mathcal{L}_{PL}$ is the language of priorities, $\mathcal{L}_{Rules}$ the language describing Priority Rules, $\mathcal{L}_{Bel}$ the language of the agent's beliefs and $\mathcal{L}_{Goal}$ that of the agent's goals. Using this $\mathcal{L}_{KR}$, the bdus for $\mathcal{L}_{Arg}$ are defined as follows:

DEFINITION 4 (BASIC DECLARATIVE UNITS FOR $\mathcal{L}_{Arg}$). *Let $\delta \in \mathcal{L}_{Rep}$ be an agent's trust evaluation based on inference rules $\mathcal{I}^{\Psi,\gamma}$, such that $\Delta \vdash^{\Psi,\gamma} \delta$ with $\Delta \subseteq \mathcal{L}_{Rep}$, $\Psi \subseteq \mathcal{L}_{Bel}$ and $\gamma \in \mathcal{L}_{Goal}$. For each $\iota \in \mathcal{I}^{\Psi,\gamma}$, let $Params_\iota$ be the corresponding sets of parameters. Let* labels *be a function that, given a set of parameters, returns a set of constants in $\mathcal{L}_{PL}$, the language of the priority system. Additionally let $\Xi \subseteq \mathcal{L}_{Rules}$ be the agent's set of trust priority rules and $\Pi \subseteq \mathcal{L}_{PL}$ be its priority system based on Ψ and γ, then:*

1. *For any sentence $\psi \in \Psi$, there is a corresponding bdu $(\{\psi\} : \psi)$ in $\mathcal{L}_{Arg}$.*
2. *The goal γ has a corresponding bdu $(\{\gamma\} : \gamma)$ in $\mathcal{L}_{Arg}$*
3. *For all priorities $\pi \in \Pi$ and all the rules $\xi \in \Xi$ the following bdus are generated:*
   - *if ξ has the form $\Phi \rightsquigarrow_{Belief} \pi$ and $\Phi \subseteq \Psi$ then $(\{(\bigwedge_{\varphi\in\Phi} \varphi) \rightarrow \pi\} : (\bigwedge_{\varphi\in\Phi} \varphi) \rightarrow \pi)$ is a bdu in $\mathcal{L}_{Arg}$*
   - *if ξ has the form $\gamma \rightsquigarrow_{Goal} \pi$ then $(\{\gamma \rightarrow \pi\} : \gamma \rightarrow \pi)$ is a bdu in $\mathcal{L}_{Arg}$*
4. *For all $\alpha_1, \ldots, \alpha_n$ such that $\Delta \vdash^{\Phi,\gamma} \alpha_k$ for all $k \in [1,n]$, if there exists an application of an inference rule $\iota^{\Psi,\gamma} \in \mathcal{I}^{\Psi,\gamma}$, such that $\frac{\alpha_1,\ldots,\alpha_n}{\beta}$ and $\mathsf{labels}(Params_{\iota^{\Psi,\gamma}}) = L$ then $(\{(\bigwedge_{\pi\in\Pi_L} \pi) \rightarrow (\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta)\} : (\bigwedge_{\pi\in\Pi_L} \pi) \rightarrow (\alpha_1 \wedge \cdots \wedge \alpha_n \rightarrow \beta))$ is a bdu of $\mathcal{L}_{Arg}$. With $\Pi_L \subseteq \Pi$ the set of priorities corresponding to labels L.*

In items 1 and 2 the relevant elements of the agent's reasoning are added to the argumentation language. In items 3

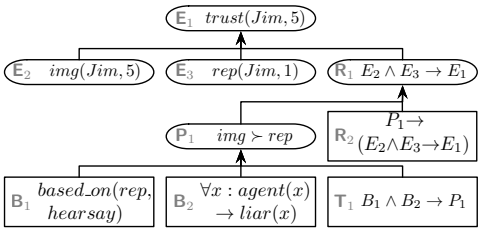


Figure 1: An example of an argument. The rectangular nodes are bdus.

and 4 the implements for reasoning about trust are added: in 3 the trust priority rules of AdapTrust, which link beliefs and goals to priorities, and in 4 the rules of the trust model. The bdus added in 4 contain a double implication: they state that if an agent has the priorities in Π_L then a *trust rule* (which was a bdu in Pinyol's argumentative theory) holds. In practice what this accomplishes, is to allow the argumentation to go a level deeper: agents can now argue about *why* a trust rule, representing an application of a deduction rule in the trust model, holds. An argument for a trust evaluation can be represented in a tree. We call this an argumentation tree and give an example of one in Figure 1. The argumentation tree can be followed by applying the deduction rules of $\mathcal{L}_{Arg}$ at each level. In order to be succinct we have omitted the defeasible knowledge part of the sentences in each node. Furthermore, we use shorthand in the tree by referring to nodes, rather than repeating the content of a node. For instance in node R_1 we can expand $E_2 \wedge E_3 \rightarrow E_1$ to its meaning: $img(Jim, 5) \wedge rep(Jim, 1) \rightarrow trust(Jim, 5)$. An argumentation tree, such as this one, is used in a dialogue to communicate personalized trust evaluations.

5.2 Dialogue Protocol for Personalizing Trust

The argumentation in the previous section can be used by an individual agent to construe the reasons for having a trust evaluation in a language that the other agents understand. We now specify a protocol that allows agents to argue back and forth in such a way that an agent is assured that, if the dialogue completes successfully, it receives a personalized recommendation at the end. The protocol is summarized in the diagram of Figure 2.

The protocol defines a dialogue for two agents; a recommendation-seeker and a recommendation-supplier. If, at any point either of the agents does not want to continue conversing, it may end the dialogue immediately. If this happens, the seeker can use any information it has obtained during the dialogue, but there is no guarantee the trust evaluations communicated use the seeker's criteria for calculating trust. In the rest of this section, we describe the other options both participants have at each point in the dialogue.

The dialogue starts with the seeker contacting the supplier to request its recommendation of a partner to achieve the seeker's goal. The supplier provides a recommendation, at which point the dialogue begins in earnest. The guiding principle in the dialogue is that the seeker agent is trying to decide whether the recommendation is acceptable or what further information and adaptation is required for this. Thus, in the diagram of Figure 2 the first decision is whether or not to accept the argument. If the argument is not immediately accepted, or rejected, the next step is to decide which of the nodes of the argumentation tree is most likely to expedite this decision. This choice is made in the "Select node in argument" action of the diagram. In the description of the

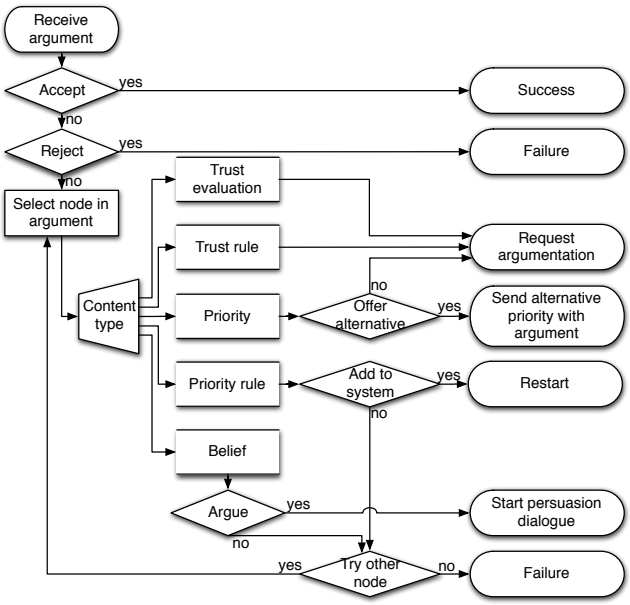


Figure 2: Diagram of the dialogue protocol

protocol below, we also describe this selection process. After selecting a node, the protocol determines what courses of action are available to an agent, based on the type of the node.

The example we use to describe the dialogue is the same as in the previous section, with the argumentation tree in Figure 1. The supplier does not reveal the entire argumentation tree at once. It only discloses information when the seeker asks for it. At the start of the dialogue, the supplier provides its trust evaluation E_1 and the direct reasons for having this evaluation (E_2, E_3 and R_1).

The seeker receives the initial argument. In order to decide whether it can accept the trust evaluation, it must decide whether it can accept the premises of the argument. This decision depends on the type of premises.

In $\mathcal{L}_{Arg}$, a trust evaluation is based on a trust rule and a number of inputs for the trust model. In the example these are trust rule R_1 and the trust evaluations E_2 and E_3. To decide whether or not to accept a trust rule, the seeker can compare it to the output of its own trust model, by using this with the inputs in the argument. In the example, the seeker can use trust evaluations E_2 and E_3 as inputs in its own trust model: if the output is equal to evaluation E_1 it accepts that the underlying criteria for the supplier's trust model are similar to the criteria in its own trust model. If not, it knows that the supplier's trust model is different from its own. The protocol gives a single course of action for trust rules: ask the supplier's reasons for it.

Another possibility is to ask about the trust evaluations used as input for the supplier's trust model. In the example these are trust evaluations E_2 and E_3. For trust evaluations, the protocol gives a single option: to ask the supplier for its reasons, which, if supplied, would expand those nodes in the argumentation tree. We omit these expansions, because the resulting subtrees are similar to the argumentation for the root, E_1. Instead we focus on the expansion of rule R_1. The seeker asks for the argument explaining why the supplier has trust rule R_1. Upon receiving this argument the seeker starts the decision process in Figure 2 again.

The reasons for a trust rule are clearly defined in $\mathcal{L}_{Arg}$. They are priorities over the criteria and a bdu that repre-

sents the dependency of the trust rule on these priorities. In the example there is only one priority, P_1, that influences the calculation of a trust evaluation from reputation and image. The first step in the protocol is once again to decide whether to accept or reject the argumentation, this time supporting node R_1. The seeker's trust model provides a way of deciding to reject the argument: if instantiating its trust model with priority P_1 does not allow it to compute E_1 from E_2 and E_3, then the agents' underlying algorithmic methods are too dissimilar for the supplier to be able to provide a personalized recommendation. Despite both agents using the same priorities to instantiate the parameters of their trust models, they compute different evaluations from the same input. In this case the dialogue ends in failure: the seeker should reject recommendations from the supplier and try another agent. Just as in Pinyol's framework, this is still useful information: the agents know that they disagree and that, in this situation, agreement is impossible.

If, in contrast, the seeker is able to emulate the supplier's trust calculation by using its priorities, then the only possible reason to not accept the trust rule outright is because the seeker disagrees with at least one of the priorities in the argumentation. The seeker can select such a priority and choose what to do. The protocol offers two options. The first is to ask *why* the supplier believes a priority holds. The second is to propose using its own priority instead. Note that the protocol allows an agent to explore both possibilities: if at a later stage it reaches "Try other node" it can try the alternative approach. The example continues using the former option for the only priority available, P_1, but the latter approach is equally valid and is described in Section 5.2.2.

5.2.1 Reasoning about the supplier's priorities

If the seeker asks why the supplier believes a priority holds, the dialogue continues. In the example, the reasons for the supplier having priority P_1 are in the 4th level of the argumentation tree of Figure 1.

The reasons for prioritizing one criterion over another, are given by the priority rules of AdapTrust, which are adopted as bdus in $\mathcal{L}_{Arg}$. These priority rules are supported by either beliefs or a goal. If the priority is supported by beliefs, as in the example, the protocol defines four possibilities:

1. The seeker chooses not to add the priority rule to its system. In this case its trust model will continue to be based on different criteria from the supplier's. It can try to convince the supplier to use its own priorities instead.

2. The seeker agent tries to add the priority rule to its system. This rule does not conflict with the rules it already knows. In this case it can be seen as a gap in the agent's knowledge and it can choose to adopt this rule.

3. The seeker agent tries to add the priority rule to its system and this rule does conflict with the rules it holds. In this case the agents have found a context in which agreement is impossible: the cognitive underpinnings of their trust models are different in this situation. The seeker agent should reject recommendations from the supplier in this context.

4. The agents enter a separate persuasion dialogue in order to convince each other about the validity of their beliefs. This can be done using a state-of-the-art argumentation framework for persuasion, such as the one proposed in [15].

Priority rules can also have goals in the antecedents, which are treated similarly, although the option for a persuasion dialogue is then not present. Conflicts between priority rules are defined as follows:

Definition 5 (Conflict of Priority Rules). *Let $U \subseteq \mathcal{L}_{Rules}$ be a set of priority rules such that:*

1. $\Pi = \{\pi' | (\Phi' \rightsquigarrow_{beliefs} \pi') \in U\}$ *is satisfiable in* $\mathcal{L}_{PL}$
2. *the set Φ, defined as the union of all Φ', such that $(\Phi' \rightsquigarrow_{Belief} \pi') \in U$, is satisfiable in* $\mathcal{L}_{Bel}$

Then a priority rule $\Psi \rightsquigarrow_{Belief} \pi$ conflicts directly with U iff $\Pi \cup \{\pi\}$ is unsatisfiable and $\Phi \models \Psi$.

A set of priority rules $\Xi \subseteq \mathcal{L}_{Rules}$ conflicts with a rule ξ if there is a set $U \subseteq \Xi$ that conflicts directly with ξ.

Note that two rules do not conflict if their antecedents are merely consistent, but only if one follows directly from the other. This is because two consistent antecedents with different conclusions might be designed to trigger in different situations, which is after all dependent on the beliefs and goals an agent has. In the case of two rules with conflicting conclusions triggering, AdapTrust contains a mechanism for choosing a consistent set of priorities. Definition 5 only defines conflicts for priority rules over beliefs. For goals it is the same, but then it is simply that a single goal leads to a conflicting set of priorities.

5.2.2 Reasoning about the seeker's priorities

If, instead of continuing the argument about the supplier's priority, the seeker proposes an alternative priority, the roles in the dialogue are switched. Now the supplier needs to discover why it should accept the seeker's priority. The same decision tree, in Figure 2, is used, but now the supplier performs the choices. Note that there are always less options, because using our $\mathcal{L}_{Arg}$, the reason for having a priority cannot be a trust evaluation, or a trust rule. Note that the supplier has the possibility to accept a priority rule into its knowledge base, but, unlike the seeker, can do this only temporarily: it may do this with the sole purpose of calculating a personalized trust evaluation for the seeker and its goal.

If at any point in the dialogue, either agent has adapted its trust model, they should restart the dialogue in order to verify that they have reached agreement and the supplier is able to provide personalized recommendations.

6. EXPERIMENTS

In the previous section we described a new argumentation framework to be able to communicate personalized trust evaluations. We now compare this model of communication to Pinyol's argumentation framework [14]. We have implemented AdapTrust using Jason [3]. In order to make a fair comparison, we keep everything as similar as possible to the experimental evaluation in [14], so we use the trust model Repage [18] and run the experiment in a simulated eCommerce environment, in which we evaluate the accuracy of buyers' trust evaluations of the sellers by using three methods of communication: (1) accepting other agents' trust evaluations directly (no argumentation), (2) filtering out mismatched communication with argumentation (Pinyol's argumentation) and (3) our model for communicating personalized trust evaluations.

6.1 The Simulation Environment

The simulation environment initially runs 20 agents who need to buy an item from any one of the 40 sellers in the environment, as in Pinyol's simulation. The sellers in this environment offer products with a constant price, quality

and delivery time. These aspects of the product are used to evaluate the trustworthiness of the seller. A buyer can be "frugal", in which case it gives more importance to the quality of the product than to the price or delivery time. A buyer can also be "stingy", in which case it evaluates price as being more important than delivery time or quality. Finally, a buyer can be "impatient", in which case the delivery time is the most important. The buyer profiles are implemented using AdapTrust's priority rules, based on the beliefs of the agent.

In addition to these basic profiles, the buyers can have different goals. We have implemented the goal to buy a bicycle, which is not associated with any priority rules, and the goal to buy milk, which must be delivered quickly and thus has an associated priority rule to prioritize delivery time over both quality and price.

These two types of priority rules and the different profiles and goals of the agents allow them to benefit from the full dialogue of Section 5.2. Agents can attempt to persuade each other to switch their basic profile. Because we rely on pre-existing persuasion dialogues for this, we have simply hard-coded the outcome. A frugal agent can persuade a stingy agent to change its profile (and thus become frugal as well): a good quality item allows one to save money in the longer term by not needing to replace it as soon. This serves both agents' underlying motivation of saving money. Furthermore, the different goals, and associated priority rules allow recommendation-suppliers to personalize their recommendation to the seeker's goal, as well as have the agents exchange priority rules for their goal.

The simulation environment runs for 40 rounds to initialize the environment. In each round the buyers buy an item from a random seller. To ensure that no single buyer can obtain full knowledge, by interacting with all the sellers, each buyer can only interact with a configurable percentage of the sellers. This percentage is thus a control on the amount of knowledge each individual buyer can obtain about the set of sellers. After buying, the buyers can communicate their trust evaluations to exactly one other buyer. Depending on the type of communication we wish to evaluate, they use no argumentation, Pinyol's argumentation, or personalized trust evaluations to perform this communication.

After this initialization, we create a new agent, which is the one to be evaluated. This agent knows nothing about the environment. It is a frugal agent with a 50/50 chance to have either goal, to buy a bicycle or milk, the same as the other buyer agents in the system. However, this agent does not explore by interacting with random sellers, but rather needs to discover the sellers' trustworthiness through communication with the established buyers. For this, it uses the configured communication model, no argumentation, Pinyol's model, or ours.

6.2 Simulation Results

The results are plotted in Figure 3. The experiments were run with an equal distribution of sellers offering one of either good quality, price or delivery time. Similarly the buyers were equally distributed over frugal, stingy and impatient agents. The experiment agent was always frugal and had a 50/50 chance of having the goal to buy a bicycle or milk. On the x-axis is plotted the percentage of sellers each buyer can interact with directly during the initialization and is thus a measure of the knowledge each agent can obtain about the

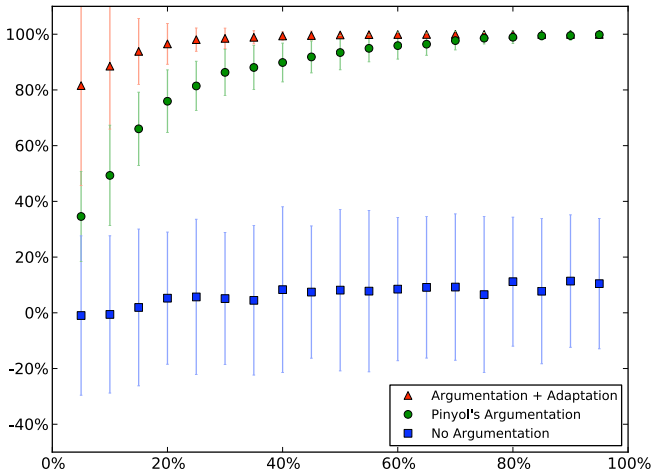


Figure 3: Experimental results. The x-axis represents the knowledge in the system and the y-axis the quality of the evaluation.

environment. With 20 buyers, 5% is the minimum to have all the sellers covered by at least one buyer. In this case, to obtain information about all sellers, information from all the buyers is needed. As the percentage of sellers each buyer can interact with increases, it becomes easier to obtain an accurate evaluation through communication, because the experiment agent needs to communicate with less of the established buyers to cover all the sellers. The y-axis plots the average accuracy of the evaluation of all the sellers in the system. After the experiment agent has finished its communication, it gives its evaluation of each of the seller agents. This "estimate" is compared to what its evaluation would be if it interacted with that seller. The difference of these two evaluations is the error of the experiment agent and we take the average of these errors as its score. To convert this to a percentage we compare this error to the expected error, if both the estimate and actual evaluations were chosen at random. This is equal to the expected difference between two standard uniform distributions, which is $\frac{1}{3}$. The y-axis thus plots the percentage increase in accuracy over this expected error of a random evaluation. Each point is the average of 100 experiments with the error bar being 1.96 standard deviations (representing an expected 95% of the population).

7. DISCUSSION

The experiment in the previous section is a proof-of-concept demonstration of the presented method of personalized trust communications. Despite being a prototypical implementation, the experiment already displays some interesting features of this method. Our method displays the greatest gains over Pinyol's argumentation in scenarios where each individual agent has little information about the entire scenario. When the amount of information available to each buyer is high, both Pinyol's and our own method can obtain near perfect information, because an agent can afford to discard information from a greater number of agents. However, when buyers do not have a lot of information, it is necessary to obtain information from a larger number of agents to accurately assess the trustworthiness of the sellers in the system. When buyers can interact with 20% of the sellers, our method is still slightly over 20% more accurate than Pinyol's method in the experiment scenario. We feel this increase justifies the greater complexity and communication

of our method. Note that agents having had direct interactions with 20% of the providers of a service is already on the high side for many eCommerce, P2P or grid computing scenarios. Despite this, we do not claim that the results from this experiment carry over to other scenarios. We run the experiment with a uniform distribution of both sellers' qualities and buyers' criteria in a simplified representation of an eCommerce environment. Even in this simple environment, if we change the parameters, we see different results. Specifically, the less likely it is that the experiment agent finds agents who are like-minded, the more important it becomes for it to obtain personalized trust recommendations from agents whose evaluation would otherwise need to be discarded. More experimentation is needed in more diverse scenarios to decide when personalized communication about trust offers useful benefits to the agents. This experiment's purpose is to demonstrate the method's functional viability and sketch the general domain in which we expect agents could use it.

7.1 Applications

Despite the experiment being based on a small and simulated scenario, it is able to show that even with just three parameters for the agents and two different goals, a filtering method will be left with too little information to work with, necessitating the use of a method such as the one proposed in this paper. This simple scenario serves as a proof of concept for its application in more realistic scenarios, such as the following:

P2P routing problems – one of the problems encountered in P2P networks is that of routing information. Deciding which peer can be trusted to transfer the required information does not have a trivial answer, especially if the network is used for diverse purposes, such as streaming different types of media, for which different agents have different requirements. Current trust and reputation models offer a possible solution [11], but only if they can get enough information. Because the environment is generally considered open and highly dynamic, exchanging information is a necessity for trust models to work, and our method provides this.

Automated eCommerce agents – the scenario we presented in the experimentation was a simplified eCommerce environment, but as the scenario is extended with more items and more properties of these items, the probability of coinciding with another agent decreases correspondingly. Therefore, despite there also being a far larger number of agents in the system, those with similar backgrounds to the own will still be sparse, necessitating a communication model such as the one we describe. If the community of sellers and buyers is relatively stable, then it might be possible to use a translation approach, as described in [8], but if this is not the case then our method provides a solution.

7.2 Future work

We recognize that the method we presented requires more extensive experimental evaluation. We regard such an evaluation as future work and intend to use personalized trust communication in different, realistic, scenarios and compare it to contemporary recommender systems or the use of reputation to give a more precise indication of what applications will truly benefit from this model. The permitted options in the dialogue can also be extended, for instance with persuasion about the correctness of priority rules. We intend to provide a formal dialogue protocol and include such extensions in the near future.

Acknowledgements

This work is supported by the Generalitat de Catalunya grant 2009-SGR-1434, the Agreement Technologies project (CONSOLIDER CSD2007-0022, INGENIO 2010) and the CBIT project (TIN2010-16306).

8. REFERENCES

[1] L. Amgoud and H. Prade. Using arguments for making and explaining decisions. *Artificial Intelligence*, 173(3–4):413–436, 2009.

[2] T. J. M. Bench-Capon. Persuasion in practical argument using value-based argumentation frameworks. *JLC*, 13(3):429–448, 2003.

[3] R. Bordini, J. Hübner, and M. Wooldridge. *Programming MAS in AgentSpeak using Jason*. Wiley, 2007.

[4] C. Chesñevar and G. Simari. Modelling inference in argumentation through labelled deduction: Formalization and logical properties. *Logica Universalis*, 1(1):93–124, 2007.

[5] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 7(2):321–358, 1995.

[6] E. Erriquez, W. van der Hoek, and M. Wooldridge. An abstract framework for reasoning about trust (extended abstract). In *AAMAS'11*, pages 1085–1086, 2011.

[7] N. D. Jones. *Computability and Complexity: From a Programming Perspective*. MIT Press, 1997.

[8] A. Koster, J. Sabater-Mir, and M. Schorlemmer. Inductively generated trust alignments based on shared interactions (extended abstract). In *AAMAS'10*, pages 1571–1572, 2010.

[9] A. Koster, M. Schorlemmer, and J. Sabater-Mir. Opening the black box of trust: Reasoning about trust models in a BDI agent. *JLC*, In Press.

[10] S. Parsons, Y. Tang, E. Sklar, P. McBurney, and K. Cai. Argumentation-basded reasoning in agents with varying degrees of trust. In *AAMAS'11*, pages 879–886, 2011.

[11] A. Perreau de Pinninck, M. Schorlemmer, C. Sierra, and S. Cranefield. A social network defence against white-washing. In *AAMAS'10*, pages 1563–1564, 2010.

[12] I. Pinyol and J. Sabater-Mir. Towards the definition of an argumentation framework using reputation information. In *Proc. of TRUST@AAMAS'09*, pages 92–103, 2009.

[13] I. Pinyol and J. Sabater-Mir. An argumentation-based protocol for social evaluations exchange. In *ECAI'10*, Lisbon, Portugal, 2010.

[14] I. Pinyol Catadau. *Milking the Reputation Cow: Argumentation, Reasoning and Cognitive Agents*, volume 44 of *Monografies de l'IIIA*. CSIC, 2011.

[15] H. Prakken. Models of persuasion dialogue. In I. Rahwan and G. Simari, editors, *Argumentation in Artificial Intelligence*, chapter 14, pages 281–300. Springer, 2009.

[16] A. S. Rao and M. P. Georgeff. Modeling rational agents within a BDI-architecture. In *Proc. of KR'91*, pages 473–484. Morgan Kaufmann, 1991.

[17] F. Ricci, R. Lior, B. Shapira, and P. B. Kantor. *Recommender Systems Handbook*. Springer, 2010.

[18] J. Sabater-Mir, M. Paolucci, and R. Conte. Repage: REPutation and imAGE among limited autonomous partners. *JASSS - Journal of Artificial Societies and Social Simulation*, 9(2), 2006.

[19] W. Teacy, J. Patel, N. Jennings, and M. Luck. TRAVOS: Trust and reputation in the context of inaccurate information sources. *JAAMAS*, 12(2):183–198, 2006.

From axiomatic to strategic models of bargaining with logical beliefs and goals

Quoc Bao Vo and Minyi Li
Faculty of Information & Communication Technologies
Swinburne University of Technology, Australia
email: {bvo | myli}@swin.edu.au

ABSTRACT

In this paper, we introduce axiomatic and strategic models for bargaining and investigate the link between the two. Bargaining situations are described in propositional logic while the agents' preferences over the outcomes are expressed as ordinal preferences. Our main contribution is an axiomatic theory of bargaining. We propose a bargaining solution based on the well-known egalitarian social welfare for bargaining problems in which the agents' logical beliefs specify their bottom lines. We prove that the proposed solution is uniquely identified by a set of axioms. We further present a model of bargaining based on argumentation frameworks with the view to develop a strategic model of bargaining using the concept of minimal concession strategy in argument-based negotiation frameworks.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: *Multiagent Systems, Intelligent Agents*; J.4 [**Computer Applications**]: *Social and Behavioral Sciences*

General Terms

Design, Economics

Keywords

Bargaining and negotiation, Collective decision making, Judgment aggregation and belief merging

1. INTRODUCTION

The formal theory of bargaining originated with John Nash's seminal papers [10, 11]. Nash's 1950 paper establishes a framework for bargaining analysis. In this paper, Nash initiated an *axiomatic approach* to bargaining, in which we abstract from the bargaining process itself and specify a list of properties (*axioms*) that a bargaining solution should satisfy. Nash then proposed four axioms and proved that they uniquely characterise what is now known as the Nash bargaining solution. In Nash's 1953 paper, he then turned to the question of how this solution might be obtained in bargaining situations between self-interested agents; i.e., investigate the bargaining problem using a strategic approach. In this paper, Nash implicitly established a new research agenda, attempting to utilise the strategic (non-cooperative) approach to provide the foundations for cooperative bargaining solution concepts.[1] His approach was to design a non-cooperative game, now known as the Nash demand game, in which the only equilibrium outcome is exactly the one suggested by Nash solution.

We'll now turn our attention to multiagent systems in which agents hold beliefs about the environment they are operating in and have goals they want to achieve or maintain. In many multiagent frameworks, the agents beliefs and goals are represented as logical sentences. Moreover, the agents are required to interact, coordinate and in most cases, negotiate to reach agreements about who do what and who get what. Given that Nash's theories and most literature on bargaining have been based exclusively on utility theory (and possibly probability theory when uncertainty is present), it has been a challenge to apply these game-theoretic models to develop solutions for or analyse the agent negotiation problems when agents hold logical beliefs and goals. Attempts have been made to convert agents' goals to a form of utility through a cost function (see e.g., [13]). However, questions remain on how agents' logical beliefs can be integrated into such a framework. Other researchers have attempted to revive Nash's approaches, particularly the axiomatic approach, by studying the properties that a bargaining solution (of a bargaining problem with logical goals) should satisfy (see e.g., [9, 8, 16, 15] and the reference therein). While [9] and [8] study a number of logical properties for negotiation, it's not clear what bargaining solution they would suggest for self-interested bargainers. In [16], an interesting bargaining solution is proposed together with a study on a number of game-theoretic properties the proposed solution satisfies. Zhang [15] introduces a framework that is perhaps closest to what Nash intended with his axiomatic approach. In this paper, Zhang proposes a solution, which he calls *simultaneous concession solution*, and shows that it is exactly characterised by a set of axioms. Zhang's bargaining solution is, however, quite problematic because: (i) it's syntax sensitive and, as a consequence, prone to manipulation; and (ii) it actually removes the goals that both agents agree on, the so-called "drowning effect".

The present paper is an effort to reopen the Nash program, particularly for agent-based bargaining problems in which agents' beliefs and goals are expressed as logical sentences. Towards that end, we introduce axiomatic and strategic models of bargaining. We propose a set of axioms that a bargaining solution should satisfy and introduce a bargaining solution that is exactly characterised by the proposed axioms. Our proposed bargaining solution is quite intuitive and based on the well-known egalitarian social welfare. Subsequently, we also present a strategic model of bargaining based on the minimal concession strategy and shown that its equilibrium

[1]This research agenda has been commonly referred to as the Nash program (see [2]).

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

outcomes turns out to be the solution outcomes described by our axiomatic theory.

The paper is organised as follows: we present some technical definitions in Section 2. The axiomatic model of bargaining is introduced in Section 3. In particular, we consider two cases: when the agents' bottom lines are fixed and based entirely on the agents' initial beliefs; and when the agents' bottom lines can be revised as the negotiation progresses and the opponent can introduce convincing arguments to challenge the agent's initial bottom line. A main result is also introduced in Section 3. Section 4 presents a strategic bargaining model which is based on the minimal concession strategy with some substantial modification to allow the agents to hold their position without having to make a concession through the use of sufficiently convincing arguments. We follow the anonymous reviewers' recommendations by: (i) omitting a number of preliminary results in Section 4, replacing them instead by a discussion about our model and solution; and (ii) providing the proofs for all of the theoretical results present in this paper. We agree with the reviewers that the revised paper is more self-contained and thus, significantly improved.

2. BACKGROUND

2.1 Logical Preliminaries

We consider a propositional language $\mathcal{L}$ defined from a finite (and non-empty) alphabet $\mathcal{P}$ together with the standard logical connectives, including the Boolean constants $\top$ and $\bot$. Furthermore, we also assume that $\mathcal{P}_O \subseteq \mathcal{P}$ is the non-empty alphabet for the negotiation outcomes. That is, the propositional variables from $\mathcal{P}_O$ constitute the issues to be settled by the negotiating agents. An interpretation ω is a total function from $\mathcal{P}$ to $\{\top, \bot\}$. An interpretation ω is a model of a set of sentences $\Phi \subseteq \mathcal{L}$ if and only if every sentence in Φ is satisfied by ω. $[\![\Phi]\!]$ denotes the set of models of the set of $\mathcal{L}$-sentences Φ. Given a sentence $\phi \in \mathcal{L}$, we'll also write $[\![\phi]\!]$ instead of $[\![\{\phi\}]\!]$. An outcome (or alternative) o is a total function from $\mathcal{P}_O$ to $\{\top, \bot\}$. We denote by $\mathcal{PO}$ the set of all possible outcomes. We will identify each outcome o with the canonical term on $\mathcal{P}_O$ which has o as its unique model. For instance, if $\mathcal{P}_O = \{p, q\}$ and $o(p) = \top, o(q) = \bot$, then o is identified with the term $p \wedge \neg q$ (or $p\bar{q}$, or $\{p, \neg q\}$).

2.2 Nash bargaining theory

Nash [10] established a framework to study bargaining. In his framework, a set of bargainers $N = \{1, 2\}$ tries to come to an agreement over a set of possible alternatives A. If they fail to reach an agreement, a disagreement event D occurs. Each agent $i \in N$ has a von Neumann - Morgenstern utility function $u_i : A \cup \{D\} \rightarrow \mathbb{R}$ from which the set of all utility pairs that result from some agreement,

$$\mathcal{S} = \{(u_1(a), u_2(a)) \in \mathbb{R}^2 \ : \ a \in A\},$$

as well as the pair $d = (d_1, d_2)$, where $d_i = u_i(D)$ can be constructed.

Nash then defined the pair $(\mathcal{S}, d)$ to be a bargaining problem. Nash subsequently defined the bargaining solution to be a function $f : \mathcal{B} \rightarrow \mathbb{R}^2$ that specifies, for each bargaining problem $(\mathcal{S}, d)$, a unique outcome $f(\mathcal{S}, d) \in \mathcal{S}$.

In the same paper, Nash also introduced an axiomatic theory of bargaining. Rather than specifying an explicit model of the bargaining procedure, the axiomatic approach aims to impose properties that one wants a bargaining solution to satisfy, and then look for solutions with these properties. Nash proposed the following four axioms:

A1. *Invariance to equivalent utility representations.* Let the bargaining problem $(\mathcal{S}', d')$ be obtained from $(\mathcal{S}, d)$ by the transformations $s'_i = \alpha_i s_i + \beta_i$ and $d'_i = \alpha_i d_i + \beta_i$, where $\alpha_i > 0$, then $f_i(\mathcal{S}', d') = \alpha_i f_i(\mathcal{S}, d) + \beta_i$, for $i = 1, 2$.

A2. *Symmetry.* If the bargaining problem $(\mathcal{S}, d)$ is symmetric (i.e., $d_1 = d_2$ and $(s_1, s_2) \in \mathcal{S} \Leftrightarrow (s_2, s_1) \in \mathcal{S}$), then $f_1(\mathcal{S}, d) = f_2(\mathcal{S}, d)$.

A3. *Independence of irrelevant alternatives.* If $(\mathcal{S}, d)$ and $(\mathcal{S}', d)$ are bargaining problems such that $\mathcal{S} \subseteq \mathcal{S}'$ and $f(\mathcal{S}', d) \in \mathcal{S}$ then $f(\mathcal{S}', d) = f(\mathcal{S}, d)$.

A4. *Pareto efficiency.* If $(\mathcal{S}, d)$ is a bargaining problem with $s, s' \in \mathcal{S}$ and $s'_i > s_i$ for $i = 1, 2$, then $f(\mathcal{S}, d) \neq s$.

Another important contribution in Nash's seminal paper is that, he also tied the axiomatic theory of bargaining and his proposed bargaining solution up nicely by proving that the proposed axioms uniquely characterise the Nash bargaining solution.

3. A LOGICAL MODEL OF BARGAINING

Consider a finite set $N = \{1, 2\}$ of two agents who try to come to an agreement over the alternatives in $\mathcal{O} \subseteq \mathcal{PO}$. Each agent i has a preference relation $\succeq_i$, which is assumed to be a total pre-order (i.e., total, reflexive and transitive), defined over the set of alternatives $\mathcal{O}$. Each agent i also maintains a set of beliefs $B_i \subseteq \mathcal{L}$. Essentially, B_i represents agent i's beliefs about her available options outside of the negotiation or simply her reservation value.

REMARK: It's important to note that the agents' beliefs don't encode their hard constraints. If an agent has a hard constraint that can never be violated and that rules out an outcome $o \in \mathcal{PO}$ then $o \notin \mathcal{O}$. Rather, the agent's beliefs B_i encode her bottom line in the bargaining, in the sense that, due to the requirement of individual rationality, she will never agree on an outcome that is worse than her bottom line.

We now define the bargaining problem to be a tuple $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$, where $\mathcal{O} \subseteq \mathcal{PO}$ is the set of outcomes and B_i and $\succeq_i$ are agent i's beliefs and preference relation over $\mathcal{O}$, respectively. Following Nash [11] and other researchers in the literature of (cooperative) game-theoretic bargaining (see e.g., [12]), we are also interested in a *bargaining solution*, by which we mean a function f that specifies a unique outcome set $f(BP) \in 2^{\mathcal{O}}$ for every bargaining problem BP. The reason why we target an outcome set as the solution of the bargaining problem instead of an outcome in $\mathcal{O}$ will become clear later.

EXAMPLE 1. *A vendor agent of a house (agent 1) is negotiating with a prospective buyer (agent 2) over the sale of the house. The two issues they need to come to an agreement are: the price of the house (i.e. whether the buyer should pay the vendor's asking price), a proposition denoted by P, and the settlement (i.e., whether it should be an early settlement), denoted by E. Then $\mathcal{O} = \{PE, P\bar{E}, \bar{P}E, \bar{P}\bar{E}\}$ is the set of possible outcomes. In this example, an outcome, say $\bar{P}\bar{E}$, indicates that the buyer would pay a price lower than the vendor's asking price such as the median house price of the area and the settlement will be according to the standard settlement of three months after the date of purchase. Assume further that F denotes the proposition that the vendor agent has an existing offer agreeing to pay him the asking price, and A denotes the proposition that the buyer can get a similar property at a price lower than the asking price. Given: $B_1 = \{F, F \Rightarrow P, A \Rightarrow \neg P\}$ and $B_2 = \{F \Rightarrow P, A \Rightarrow \neg P\}$,*

and $PE \succ_1 P\bar{E} \succ_1 \bar{P}E \succ_1 \bar{P}\bar{E}$ *and* $\bar{P}\bar{E} \succ_2 P\bar{E} \succ_2 \bar{P}E \succ_2 PE$*, a bargaining problem* $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ *can be defined.*

REMARK: Our bargaining model defined above is quite similar to the ordinal bargaining model defined by Shubik [14]. In this model, a bargaining situation (between two players) can be represented as a tuple $(A, D, \geq_1, \geq_2)$, where A is the set of possible agreements, D is the disagreement (i.e., the outcome when the agents fail to reach an agreement), and $\geq_1$ and $\geq_2$ are the preference orderings over the set $A \cup \{D\}$ of the agents 1 and 2, respectively.[2] In the ordinal bargaining model, a bargaining solution F maps from every bargaining situation $(A, D, \geq_1, \geq_2)$ to an agreement $f(A, D, \geq_1, \geq_2) \in A$ (see e.g., [12] for more details).

In the following, we'll explore the cases when the agents' beliefs define their bottom-lines (and thus, the disagreement position of the agents), and also when the agents' beliefs are uncertain and can be revised (in relation to the opponents' beliefs and position).

3.1 Bargaining with fixed bottom lines

When the agents' beliefs are not revisable, they define the agents' disagreement points. We can now discuss a reformulation of Nash's axioms for the logical bargaining model by associating agent i's disagreement point with his beliefs. Firstly, we will define the disagreement point for a logical bargaining problem. Intuitively, the notion of disagreement point (or threat point) has been used to encode a bargainer's bargaining power. That is, the higher the utility of the disagreement point to a bargainer, the more power she has as she can walk away from the negotiation and obtain that high utility outside of the negotiation. This matches with our designation of the agent's beliefs. It's her beliefs that define what she thinks she and her opponent can get outside of the negotiation, which subsequently defines her relative bargaining power and the negotiation disagreement point.

DEFINITION 1. *Let* $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ *be a bargaining problem, agent* i*'s* **disagreement point** *is the outcome* $D_i \in \mathcal{O}$ *such that (i)* D_i *is least preferred to* i*, and (ii)* D_i *is consistent with* B_i*. Then, agent* i*'s* **bargaining power** *is defined to be the number of outcomes ruled out by* D_i*, according to agent* i*'s preference ordering:*

$$U_i(D_i) = \#\{o \in \mathcal{O} : D_i \succ_i o\}.$$

An outcome $o \in \mathcal{O}$ *is* **agreement-feasible** *if* $o \succeq_i D_i$ *for* $i = 1, 2$.

For bargaining with ordinal preference, it has been shown by Osborne and Rubinstein [12] that a reformulation of **A1** (i.e., an axiom expressing Invariance of Equivalent Preference Representations) would result in unattractive bargaining solutions.

In the rest of this paper, we'll denote the agent other than agent i by $-i$. Furthermore, for convenience of presentation, given two outcomes $o, o' \in \mathcal{O}$, we'll say that $o' \succeq o$ if and only if $o' \succeq_i o$ for $i = 1, 2$; similarly, $o' \approx o$ if and only if $o' \succeq o$ and $o \succeq o'$.

Pareto Efficiency axiom can be formulated in our model as follows.

PE. If $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ is a bargaining problem with $o, o' \in \mathcal{O}$ and $o' \succeq o$ and $o' \succ_j o$ for some $j \in \{1, 2\}$, then $o \notin f(BP)$.

Note that in this paper, by Pareto-efficiency we mean the Strong Pareto Efficiency, rather than the Weak Pareto Efficiency which states that an outcome is only inefficient when there is some other outcome that can improves for *all* agents.

Next, the axiom of Independence of Irrelevant Alternatives will be formulated in our model as follows.

IIA. If $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ and $BP' = (\mathcal{O}', \langle B_1, \succeq_1 \rangle, \langle B_2, \succeq_2\rangle)$ are bargaining problems with $\mathcal{O} \subseteq \mathcal{O}'$, and $f(BP') \subseteq \mathcal{O}$ then $f(BP) = f(BP')$.

Finally, axiom Symmetry can be interpreted as imposing the requirement of fairness on a bargaining solution. That is, when the bargaining situation is symmetric for the two bargainers in the sense that they have similar bargaining power and that there is no possible agreement that can provide a particular payoff structure to the two bargainers without another possible agreement that can provide the opposite payoff structure, then the solution should give the bargainers the same payoffs. With a discrete set of alternatives, it can not be guaranteed to have an alternative that satisfies this property.

EXAMPLE 2. *Continue with our running example, consider the following bargaining situation:* $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ *where* $B_1 = \{E\}$ *and* $B_2 = \{E\}$ *(i.e., both bargainers insist in an early settlement), and* $PE \succ_1 P\bar{E} \succ_1 \bar{P}E \succ_1 \bar{P}\bar{E}$ *and* $\bar{P}\bar{E} \succ_2 PE \succ_2 \bar{P}E \succ_2 P\bar{E}$*. It's easy to see that the two agents have similar bargaining power in which they would both rule out the two least preferred outcomes and it happens that, in this bargaining situation, they share the same set of outcomes they are willing to agree on, namely the set* $\{PE, \bar{P}E\}$*. However, they have opposite preferences over the outcomes in this set. Thus, neither outcome would be an attractive solution in this bargaining situation. By allowing this set of outcomes to be the solution in this situation, we are open to any resolution, including Nash's suggestion of non-physical agreements such as the lotteries over these outcomes.*

To ensure fairness, we will target outcomes that aim at maximising the payoffs for agents with the smallest gains (in utility). We define a cardinal gain of an outcome o for an agent i to be the number of outcomes between o and the disagreement point D_i. Formally,

DEFINITION 2. *Let* $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ *be a bargaining problem and an agreement-feasible outcome* $\sigma \in \mathcal{O}$*, the (cardinal)* **gain** *of outcome* σ *for agent* i *is defined as follows:*

$$G_i(\sigma) = \#\{o \in \mathcal{O} : \sigma \succ_i o \ \&\ o \succeq_i D_i\}.$$

Basically, the axiom for fairness, to be presented in the following, ensures that the difference between the bargainers' gains would be minimal. However, to avoid the bargainers to settle for fair but suboptimal outcomes, we require only that fairness be subject to the optimality of the outcome. We will introduce a concept to allow efficiency to be formulated in unanimity, namely *Unanimous Efficiency* (UE). Intuitively, if o' can improve for both the worst-off agent and the better-off agent in comparison to o then o is considered to be UE-dominated by o' and should not be selected as a bargaining agreement. Formally,

DEFINITION 3. *Let* $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ *be a bargaining problem with the agreement-feasible outcomes* $o, o' \in \mathcal{O}$*,* o *is* **UE-dominated** *by* o' *if* $\min_{i=1,2}(G_i(o')) \geq \min_{i=1,2}(G_i(o))$ *and* $\max_{i=1,2}(G_i(o')) \geq \max_{i=1,2}(G_i(o))$*, and at least one of them has to be a strict inequality.*

Moreover, we'll also say that two outcomes o *and* o' *are* **UE-equivalent** *if and only if* $\min_{i=1,2}(G_i(o')) = \min_{i=1,2}(G_i(o))$ *and* $\max_{i=1,2}(G_i(o')) = \max_{i=1,2}(G_i(o))$.

[2]The preference orderings are required to be total pre-orders. That is, a complete transitive reflexive binary relation.

An outcome is **unanimously efficient** *if it is not UE-dominated by any other outcome.*[3]

LEMMA 1. *Let $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ be a bargaining problem with an agreement-feasible outcome $o \in \mathcal{O}$, if o is unanimously efficient then it is also Pareto efficient.*

Before proving Lemma 1, we state a convention to be used throughout the rest of the paper: Given an outcome o, if $G_1(o) = G_2(o)$, we assume that $\arg\min_{i=1,2}(G_i(o))$ will pick out a single value, either 1 or 2 (which one would be picked doesn't matter). Furthermore, if $j = \arg\min_{i=1,2}(G_i(o))$ then $\arg\max_{i=1,2}(G_i(o)) = -j$. That is, $\arg\min_{i=1,2}(G_i(o))$ (resp. $\arg\max_{i=1,2}(G_i(o))$) is guaranteed to deterministically return a single agent j (resp. $-j$) whose cardinal gain $G_j(o)$ (resp. $G_{-j}(o)$) is smallest (resp. largest).

Proof: Suppose, to the contrary, that o is not Pareto efficient. That is, there are $o' \in \mathcal{O}$ and $j \in \{1, 2\}$ such that $o' \succ_j o$ and $o' \succeq_{-j} o$. Obviously, o' is agreement-feasible. We'll consider two cases:

Case 1: $\arg\min_{i=1,2}(G_i(o')) = \arg\min_{i=1,2}(G_i(o)) = k$. If $k = j$ then $\min_{i=1,2}(G_i(o')) > \min_{i=1,2}(G_i(o))$ and $\max_{i=1,2}(G_i(o')) \geq \max_{i=1,2}(G_i(o))$. If $k \neq j$ then $\max_{i=1,2}(G_i(o')) > \max_{i=1,2}(G_i(o))$ and $\min_{i=1,2}(G_i(o')) \geq \min_{i=1,2}(G_i(o))$. Either way, we have o is UE-dominated by o', and thus can not be unanimously efficient.

Case 2: $\arg\min_{i=1,2}(G_i(o')) \neq \arg\min_{i=1,2}(G_i(o))$. Without loss of generality, we can assume that $\arg\min_{i=1,2}(G_i(o')) = 1$ and $\arg\min_{i=1,2}(G_i(o)) = 2$. That is, $G_1(o') = \min_{i=1,2}(G_i(o'))$ and $G_2(o) = \min_{i=1,2}(G_i(o))$ and $G_2(o') = \max_{i=1,2}(G_i(o'))$ and $G_1(o) = \max_{i=1,2}(G_i(o))$. Therefore, $G_1(o) \geq G_2(o)$ and $G_2(o') \geq G_2(o)$. Also, because $o' \succ_j o$ for some $j \in \{1, 2\}$, at least one of the above inequality has to be strict. Since $o' \succ_j o$ for some $j \in \{1, 2\}$ and $o' \succeq_{-j} o$,[4] $G_1(o') \geq G_1(o)$.

Hence, $G_1(o') \geq G_1(o) \geq G_2(o)$ and $G_2(o') \geq G_1(o') \geq G_1(o)$, and at least one of the inequalities $G_1(o) \geq G_2(o)$ and $G_2(o') \geq G_1(o')$ has to be strict. In other words, $\min_{i=1,2}(G_i(o')) \geq \min_{i=1,2}(G_i(o))$ and $\max_{i=1,2}(G_i(o')) \geq \max_{i=1,2}(G_i(o))$, and at least one of these inequalities has to be strict.

Therefore, o is UE-dominated by o', and thus can not be unanimously efficient. □

The following *Fairness* axiom can now be formulated.

FR. If $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ is a bargaining problem with an agreement-feasible outcome $o \in \mathcal{O}$. If there is an agreement-feasible and unanimously efficient outcome $o' \in \mathcal{O}$ such that $|G_1(o) - G_2(o)| > |G_1(o') - G_2(o')|$ then $o \notin f(BP)$.

In other words, axiom **FR** allows us to select the fairer outcomes among those that are unanimously efficient.

In addition to the above Fairness axiom, we will also require that when outcomes are UE-equivalent, the bargaining solution will not be biased towards a particular one. The following axiom formulates this requirement for unbiasedness.

UB. If $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ is a bargaining problem with agreement-feasible and UE-equivalent outcomes $o, o' \in \mathcal{O}$. Then, $o \in f(BP)$ if and only if $o' \in f(BP)$.

Finally, we will replace the Pareto Efficiency (**PE**) axiom by a stronger one, requiring that the bargaining solution be unanimously efficient, rather than only Pareto-efficient.

UE. If $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ is a bargaining problem with the agreement-feasible outcomes $o, o' \in \mathcal{O}$ and o is UE-dominated by o' then $o \notin f(BP)$.

Obviously, by Lemma 1, if the bargaining solution f satisfies **UE** then it also satisfies **PE**. In the following, we'll develop a bargaining solution that is uniquely identified by the axioms **UE**, **FR** and **UB**. The developed solution is based on the well-known *egalitarian* solution:

THEOREM 1. *There is a bargaining solution $f^E : \mathcal{BP} \to 2^{\mathcal{O}}$ given by:*

$$f^E(\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle) = \arg\max_{o \in ES}(\max_{i=1,2} G_i(o)), \text{ where}$$

$$ES = \arg\max_{o \in AF}(\min_{i=1,2} G_i(o))$$

where $AF \subseteq \mathcal{O}$ denotes the set of agreement-feasible outcomes of BP.[5] *Then, a bargaining solution $f : \mathcal{BP} \to 2^{\mathcal{O}}$ satisfies* **UE**, **FR**, *and* **UB** *if and only if $f = f^E$.*

Before proving Theorem 1, we'll discuss a relevant result. Note that, $\approx$ is an equivalence relation on the set $\mathcal{O}$. Moreover, if we'll only focus on the set $AF \subseteq \mathcal{O}$ of agreement-feasible outcomes of BP, then AF can be reduced to a collection of equivalence classes, such that each equivalence class can be represented by a pair $(G_1(o), G_2(o))$, where o is a member of the equivalence class.

LEMMA 2. *If $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ is a bargaining problem then $f^E(BP)$ contains at most two equivalence classes of the agreement-feasible outcomes of BP. Moreover, if $f^E(BP)$ contains exactly one equivalence class then it has to be of the form (g, g) (i.e., it gives both agents the same gain); if $f^E(BP)$ contains exactly two equivalence classes then they have to be of the form (g, h) and (h, g).*

Proof (of the Lemma): Obviously, from the definition of the function f^E, there exist $k_{min}, k_{max} \geq 0$ such that for every outcome $o \in ES$, $\min_{i=1,2} G_i(o) = k_{min}$ and for every outcome $\sigma \in f^E(BP)$, $\max_{i=1,2} G_i(\sigma) = k_{max}$. Thus, when $k_{min} = k_{max}$, $f^E(BP)$ contains exactly one equivalence class that is of the form (g, g), where $g = k_{min} = k_{max}$.

Otherwise, $k_{min} \neq k_{max}$ and $f^E(BP)$ contains exactly two equivalence classes of the form (g, h) and (h, g) where $g = k_{min}$ and $h = k_{max}$. □

Proof (of the Theorem): First we prove that, f^E satisfies **UE**, **FR**, and **UB**.

That f^E satisfies **UE**: Let $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ be a bargaining problem. Let $\sigma \in f^E(BP)$, we'll prove that σ is unanimously efficient. Suppose, to the contrary, that there is an agreement-feasible outcome $o \in AF$ such that $\min_{i=1,2} G_i(o) \geq \min_{i=1,2} G_i(\sigma)$ and $\max_{i=1,2} G_i(o) \geq \max_{i=1,2} G_i(\sigma)$ and at least one of these inequalities is strict. Moreover, there exists $k_{min} \geq 0$ such that $\forall a \in AF. \min_{i=1,2} G_i(a) \leq k_{min}$ and $k_{min} = \min_{i=1,2} G_i(\sigma)$. Thus, $k_{min} = \min_{i=1,2} G_i(o) = \min_{i=1,2} G_i(\sigma)$. In other words, $o \in ES$. However, according to Lemma 2, there exists $k_{max} \geq 0$ that, together with k_{min}, characterises the equivalence classes defining $f^E(BP)$ and $\forall a \in ES. \max_{i=1,2} G_i(a) \leq k_{max}$ and $k_{max} = \max_{i=1,2} G_i(\sigma)$. Thus, $\max_{i=1,2} G_i(a) \leq \max_{i=1,2} G_i(\sigma)$. Contradiction.

[3]Note that, similar to our remark about Pareto-efficiency, our definition of Unanimous Efficiency is also a strong one.

[4]We use $-j$ to denote the agent other than j.

[5]Note that more precise notations would be AF_{BP} and ES_{BP}. But since the bargaining problem BP is always clear from the context, we'll drop these subscripts.

That f^E satisfies **FR**: Suppose, to the contrary, that there is a bargaining problem $BP = (\mathcal{O}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ and $\sigma \in f^E(BP)$ such that there is a unanimously efficient outcome $o \in AF$ and $|G_1(\sigma) - G_2(\sigma)| > |G_1(o) - G_2(o)|$.

According to Lemma 2, σ belongs to the equivalence classes characterised by the two non-negative numbers k_{min} and k_{max} (possibly equal to each other).

We have $k_{min} = \min_{i=1,2} G_i(\sigma)$ and $k_{max} = \max_{i=1,2} G_i(\sigma)$.

Case 1: $\min_{i=1,2} G_i(o) = k_{min}$. Since $|G_1(\sigma) - G_2(\sigma)| > |G_1(o) - G_2(o)|$, clearly $\max_{i=1,2} G_i(o) < k_{max}$, which is a contradiction because o is UE-dominated by σ.

Case 2: $\min_{i=1,2} G_i(o) < k_{min}$. Since o is unanimously efficient, it is the case that $\max_{i=1,2} G_i(o) > k_{max}$. But then $|G_1(\sigma) - G_2(\sigma)| = k_{max} - k_{min} < \max\limits_{i=1,2} G_i(o) - \min\limits_{i=1,2} G_i(o) = |G_1(o) - G_2(o)|$. Contradiction.

It's obvious that f^E satisfies **UB**.

We are now proving that a solution f satisfying **UE**, **FR**, and **UB** necessarily obtains f^E.

Suppose, to the contrary that, there exists a solution f satisfying **UE**, **FR**, and **UB** and a bargaining problem $BP = (\mathcal{O}, \langle B_1, \succeq_1 \rangle, \langle B_2, \succeq_2\rangle)$ such that $f(BP) \neq f^E(BP)$. First, we'll show that it's not possible for $f(BP) \setminus f^E(BP) \neq \emptyset$. Assume by way of contradiction that there exists an outcome $o \in AF$ such that $o \in f(BP) \setminus f^E(BP)$. Then o is unanimously efficient and is not ruled out by axiom **FR**. According to Lemma 2, the set of outcomes $f^E(BP)$ can be partitioned into equivalence classes characterised by the non-negative numbers k_{min} and k_{max} (possibly equal to each other). Clearly, $\min_{i=1,2} G_i(o) \leq k_{min}$. If $\min_{i=1,2} G_i(o) = k_{min}$ then, for o to be unanimously efficient, $\max_{i=1,2} G_i(o) \geq k_{max}$. Thus, $\max_{i=1,2} G_i(o) = k_{max}$ In other words, $o \in f^E(BP)$. Contradiction. If $\min_{i=1,2} G_i(o) < k_{min}$ then, for o to be unanimously efficient, $\max_{i=1,2} G_i(o) > k_{max}$. But then, there is a unanimously efficient outcome $\sigma \in f^E(BP)$ such that $|G_1(\sigma) - G_2(\sigma)| = k_{max} - k_{min} < \max_{i=1,2} G_i(o) - \min_{i=1,2} G_i(o) = |G_1(o) - G_2(o)|$. Thus, $o \notin f(BP)$ according to axiom **FR**. Contradiction.

That $f^E(BP) \setminus f(BP) \neq \emptyset$ follows trivially from axiom **UB** and Lemma 2. □

It's also straightforward to see that f^E satisfies axiom **IIA**.

COROLLARY 1. *The bargaining solution f^E defined in Theorem 1 satisfies* **IIA**.

So far in this section, the agent's beliefs B_i are only used to define the agent's disagreement point and don't play much role in characterising the negotiation outcome. The problem becomes more challenging when we allow the agents' beliefs to change according to the bargaining situation.

3.2 Bargaining with revisable agents' beliefs

According to the bargaining model introduced in the preceding section, when the agents' beliefs B_i define the bottom-lines D_i that result in an empty set of agreement-feasible outcomes AF (i.e., $\{o \in \mathcal{O} : o \succeq_1 D_1\} \cap \{o \in \mathcal{O} : o \succeq_2 D_2\} = \emptyset$), agreement is not possible. Nevertheless, in most negotiations, the agents' beliefs represent their inclination toward a particular position rather than unmovable. For instance, a buyer of a house may know for certain that an identical house was sold a month ago for $\$y$, and thus is not too willing to pay much more than $\$y$ for this house. However, knowing that there is no other house left in the area that he can buy, he is perhaps willing to pay more than $\$y$, if there are compelling reasons for him to do so (e.g., there are other buyers who would like to buy a house in the area). In this situation, if P denotes the proposition that the vendor agent's asking price is higher than $\$y$, then $\neg P$ doesn't necessarily define the buyer's disagreement point. It could be the case that the buyer believes in $\neg P$, but is also willing to retract this belief when learning about the scarcity of houses as well as the high demand for houses in the area.

Given that the agents' beliefs (and constraints) will play a crucial role in this model of bargaining, we will make the agents' hard constraints explicit in our bargaining model. In the rest of the paper, we will assume that the set of feasible negotiation outcomes is defined by the hard constraints $\mathcal{C}$. We denote by $\mathcal{O}_\mathcal{C}$ the set of feasible outcomes that satisfy the hard constraints $\mathcal{C}$. That is, $o \in \mathcal{O}_\mathcal{C}$ if and only if $[\![o]\!] \cap [\![\mathcal{C}]\!] \neq \emptyset$.

Given the propositional language $\mathcal{L}$ and the non-empty alphabet $\mathcal{P}_O$ for the negotiation outcomes, a bargaining problem is defined to be a tuple $BP = (\mathcal{C}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$, where $\mathcal{C} \subseteq \mathcal{L}$ is the set of hard constraints shared by all agents, $B_i \subseteq \mathcal{L}$ is the set of agent i's beliefs, and $\succeq_i$ is agent i's preference relation over the set $\mathcal{O}_\mathcal{C}$. Subsequently, the (movable) disagreement points $D_i \in \mathcal{O}_\mathcal{C}$ are defined such that D_i is least preferred to agent i and D_i is consistent with B_i. Moreover, the set of agreement-feasible outcomes AF_{BP} is defined to be $\{o \in \mathcal{O}_\mathcal{C} : o \succeq_1 D_1\} \cap \{o \in \mathcal{O}_\mathcal{C} : o \succeq_2 D_2\}$.[6] We will first state a trivial lemma:

LEMMA 3. *Let $BP = (\mathcal{C}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ be a bargaining problem, if the set $\mathcal{C} \cup B_1 \cup B_2$ is consistent then $AF_{BP} \neq \emptyset$.*

Proof: Let ω be a model of $\mathcal{C} \cup B_1 \cup B_2$. Let $o \in \mathcal{PO}$ be such that $\omega \in [\![o]\!]$. Clearly, $o \in \mathcal{O}_\mathcal{C}$ and o is consistent with both B_1 and B_2. Thus, $o \succeq_1 D_1$ and $o \succeq_2 D_2$. Thus, $o \in AF_{BP}$. □

On the other hand, when the set $\mathcal{C} \cup B_1 \cup B_2$ is inconsistent, it doesn't always mean that $AF_{BP} = \emptyset$.

EXAMPLE 3. *Continue with our running example and consider the following bargaining situation $BP = (\mathcal{C}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2 \rangle)$, where $\mathcal{C} = \{P\}$ (e.g., the agent receives the instruction from the vendor not to sell the house for less than the asking price and this is common knowledge), and $B_1 = \{E\}$ and $B_1 = \{\neg E\}$. Also, suppose that $P\bar{E} \succ_1 PE$ and $PE \succ_2 P\bar{E}$. Clearly, $\mathcal{C} \cup B_1 \cup B_2$ is inconsistent, but $D_1 = PE$ and $D_2 = P\bar{E}$, and $AF_{BP} = \{PE, P\bar{E}\}$.*

Given Lemma 3, one fairly naive idea is to perform belief merging (see e.g., [7]) with integrity constraint (i.e., merging B_1 and B_2 with the integrity constraint $\mathcal{C}$) to obtain a consistent belief base X which will be treated as the shared bottom line for both agents 1 and 2. Subsequently, the bargaining model described in Section 3.1 can be applied to characterise the negotiation outcome. Unfortunately, this straightforward idea will not work, for the simple reason that belief merging takes into account the two belief bases B_1 and B_2 when merging them with respect to the integrity constraints $\mathcal{C}$. But it fails to take into consideration the agents' preferences $\succeq_1$ and $\succeq_2$ regarding the preferred outcomes.

EXAMPLE 4. *In the running example, consider a bargaining situation in which $\mathcal{C} = \{\top\}$ (i.e., no hard constraints), $B_1 = \{P, \neg E\}$, and $B_2 = \{E\}$. Furthermore, the agents preferences are: $PE \succ_1 P\bar{E} \succ_1 \bar{P}E \succ_1 \bar{P}\bar{E}$ and $\bar{P}E \succ_2 \bar{P}\bar{E} \succ_2 PE \succ_2 P\bar{E}$ with $D_1 = P\bar{E}$ and $D_2 = PE$. Clearly, $\mathcal{C} \cup B_1 \cup B_2$ is inconsistent and most standard belief merging mechanisms (see [7]) would result in a merge belief base $X = \Delta_\mathcal{C}(B_1 \sqcup B_2) = \{P\}$. By taking X to define the common bottom line for both agents, the new disagreement points for them become $D_1' = D_1$ and $D_2' = P\bar{E}$.*

[6] When clear from the context, we will omit the subscript and write AF instead.

Clearly, this has disadvantaged agent 2. Moreover, it has also imposed agent 1's bottom line regarding attribute P on agent 2 without any reasonable justification and compensation.

On the other hand, any mechanism that searches for a negotiation outcome based only on the agents' preferences $\succeq_1$ and $\succeq_2$ without taking into account the agents' beliefs is likely to produce impractical outcomes as well. For instance, in the bargaining situation discussed at the beginning of this section, assume that $PE \succ_1 P\bar{E} \succ_1 \bar{P}E \succ_1 \bar{P}\bar{E}$ and $\bar{P}\bar{E} \succ_2 \bar{P}E \succ_2 P\bar{E} \succ_2 PE$. Assume also that the vendor's asking price is $\$x > \$y$, and $B_1 = \{P, \neg E\}$ (i.e., the vendor agent knows that the house he is selling is currently the only house for sale in the area and there are several buyers who are looking for a house in the area, while a recent government regulation requires mortgage lender to carry out a number of checks before releasing the fund for settlement), and $B_2 = \{\neg P, \neg E\}$ (i.e., the buyer knows that an identical house was sold for $\$y$ last month). As the set of agreement-feasible outcomes AF is empty, the agents need to make concessions to possibly reach an agreement. Without taking into account the agents' beliefs, the bargaining strategy of minimal concession (see [5]) suggests the following negotiation process: First, agent 2, the buyer, will make a minimal concession from its current offer of $\bar{P}\bar{E}$ to $\bar{P}E$; then, agent 1 makes a minimal concession, and accept the offer $\bar{P}E$. This outcome is certainly not justified since the right price in this case should be $\$x$ (thus, agreeing on P) while the agents can also agree on $\neg E$.

Therefore we argue that a reasonable mechanism should enable the agents to use their beliefs to make the decision on what concession to make, taking into account their preferences. Consequently, we'll investigate a strategic model for bargaining in the following section.

4. AN ARGUMENTATION-BASED MODEL OF BARGAINING

The axiomatic bargaining model introduced above is inherently static in the sense that only the outcome, and not the bargaining process, is analysed. This ensures a number of advantages such as tractability. Nevertheless, in most circumstances, it's important to study the bargaining process as well as the bargainers' strategies. For instance, we may be interested in knowing how the bargaining outcome is affected by changes in the bargaining procedure, and what would be the best strategy or decision a bargainer should take in a given situation.

The bargaining protocol to be used by the agents to reach an agreement is based on the belief negotiation models proposed by Booth [3]. In this model, the negotiation proceeds in rounds. The negotiation starts off with the initial offer profile $\vec{\Theta}^0 = (\Theta_1^0, \Theta_2^0)$, where an offer Θ_i^j is a subset of the set of feasible outcomes $\mathcal{O}$, indicating the outcomes agent i is willing to accept after j rounds of negotiation. If $\vec{\Theta}^j$ $(j \geq 0)$ is consistent then the set of agreement-feasible outcomes $AF = \Theta_1^j \cap \Theta_2^j$ is non-empty and a physical agreement can be selected from AF. If $\vec{\Theta}^j$ $(j \geq 0)$ is inconsistent then a "contest" between the agents will be carried out to select a subset of agents who are required to "make some concessions".[7] The new offer profile $\vec{\Theta}^{j+1}$ after the selected agents making the concession allows the negotiation to proceed to the next round. Under "certain predefined conditions", a disagreement is reached. Furthermore, under a monotonic concession protocol (see [13]), the new offer profile $\vec{\Theta}^{j+1}$ is required to include the previous one; i.e., $\vec{\Theta}^j \subseteq \vec{\Theta}^{j+1}$), where $\vec{S} = (S_1, \ldots, S_n) \subseteq (T_1, \ldots, T_n) = \vec{T}$ if and only if $S_i \subseteq T_i$ for all $i \in \{1, \ldots, n\}$. A more precise bargaining protocol will be introduced later in this section.

[7]The generality of this protocol allows it to encompass many common negotiation protocols including the alternating-offer protocol and the simultaneous-concession protocol.

From the discussion in the preceding section, a bargaining outcome should be based on the agents' beliefs about the bargaining situation at hand while taking into account their preferences. To formulate the idea that an agent's beliefs that define her position on certain bargaining issues can be undermined or dominated by her opponent's beliefs, we appeal to the argumentation-based framework [4, 1]. In a strategic model of bargaining, a bargaining problem $BP = (\mathcal{C}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ is based on a common language $\mathcal{L}$ and a common outcome alphabet $\mathcal{P}_O$, with the set of hard constraints $\mathcal{C}$ being common knowledge while the agents beliefs and preferences $\langle B_i, \succeq_i\rangle$ for $i = 1, 2$ are their private information. We will first reproduce some notions of argumentation theory.

DEFINITION 4. ([1]) *An* **argument** *of a set of sentences $X \subseteq \mathcal{L}$ (aka. X-***argument***) under the constraints $\mathcal{C}$ is a pair (H, h), where $h \in \mathcal{L}$ and $H \subseteq X$, satisfying:*

(i) *$\mathcal{C} \cup H$ is consistent,*

(ii) *$\mathcal{C} \cup H \models h$, and*

(iii) *H is minimal (i.e., no strict subset of H satisfies* (i) *and* (ii)*).*

H is called the **support** *and h the* **conclusion** *of the argument (H, h). Moreover, given two arguments (H, h) and (H', h'), if $H \Leftrightarrow H'$ and $h \Leftrightarrow h'$ then we treat them as the same argument. That is, a set of arguments can not contain both arguments.*

An argument (H', h') is a **subargument** *of the argument (H, h) iff $H' \subseteq H$.*

Given a set of arguments Γ, the **base** *of Γ is the set: $\mathcal{B}_\Gamma = \bigcup_{(H,h)\in\Gamma} H$.*

We denote by $\mathbb{A}_\mathcal{C}(X)$ the set of all X-arguments under the constraints $\mathcal{C}$.

DEFINITION 5. ([1]) *Let (H, h) and (H', h') be two arguments of $\mathbb{A}_\mathcal{C}(X)$:*

- *(H, h)* **rebuts** *(H', h') if and only if $h \Leftrightarrow \neg h'$.*
- *(H, h)* **undercuts** *(H', h') if and only if $h \Leftrightarrow \neg h''$ for some $h'' \in H'$.*

When (H, h) rebuts or undercuts (H', h'), we also say that (H, h) **attacks** *(H', h'). When (H, h) attacks (H', h') and (H'', h'') attacks (H, h), we say that (H'', h'')* **defends** *(H', h').*

We are now in a position to formally define our bargaining protocol. First, we define the notion of bargaining proposal.

DEFINITION 6. *Let $BP = (\mathcal{C}, \langle B_1, \succeq_1\rangle, \langle B_2, \succeq_2\rangle)$ be a bargaining problem and $\mathcal{O}_\mathcal{C}$ denote the set of feasible outcomes. A* **bargaining proposal** *(or,* **proposal***) by agent i at stage j, denoted by ρ_i^j is a pair (Θ_i^j, Γ_i^j), where $\Theta_i^j \subseteq \mathcal{O}_\mathcal{C}$ is the set of outcomes agent i is willing to agree on and $\Gamma_i^j \subseteq \mathbb{A}_\mathcal{C}(\mathcal{L})$ is the set of arguments agent i has used to defend her offers Θ_i^j.*

The pair (ρ_1^j, ρ_2^j) of the agents' proposals in stage j is called the **bargaining context** *at stage j.*

It's important to note that, in a strategic model of bargaining the agents beliefs and preferences $\langle B_i, \succeq_i\rangle$ are their private information. Therefore, agent i can introduce arguments that are not based on her beliefs if she thinks that they would give her an advantage. In the following, we'll write $\bigvee \Theta$ instead of $\bigvee_{o\in\Theta} o$. We will now define the notion of an argument being relevant to a bargaining context.

DEFINITION 7. *Let (ρ_1, ρ_2) be a bargaining context, where $\rho_i = (\Theta_i, \Gamma_i)$ is agent i's proposal, for $i = 1, 2$. An argument (H, h) is* **relevant** *to this bargaining context (for agent i) if $(H, h) \notin \Gamma_i$ and*

- $\bigvee \Theta_{-i} \wedge h \models \perp$*; or*
- (H, h) *attacks an argument* $(H', h') \in \Gamma_{-i}$.

Of the two non-trivial conditions above, the former says that agent i rejects agent $-i$'s current offered outcomes Θ_{-i} by advancing an argument (H, h) that contradicts Θ_{-i} and thus requires agent $-i$ to make a concession. The latter allows agent i to advance an argument (H, h) to defeat a relevant argument (H', h') advanced by agent $-i$ in previous rounds of bargaining.

In the bargaining protocol informally described at the beginning of this section, for the "contest" to select who need to make a revised proposal during a negotiation round, we assume that all agents will have to submit the updated proposal in each round. Furthermore, an agent i's proposal in round 0 has to contain a non-empty offer $\Theta_i^0 \neq \emptyset$ and, to simplify the protocol, it also contains an empty set of arguments $\Gamma_i^0 = \emptyset$. Agent i's proposal in round $j > 0$, $\rho_i^j = (\Theta_i^j, \Gamma_i^j)$ is required to meet the following conditions:

- $\Theta_i^j \supseteq \Theta_i^{j-1}$;
- $\Gamma_i^j \supseteq \Gamma_i^{j-1}$ such that $\mathcal{B}_{\Gamma_i^j} \cup \mathcal{C}$ is consistent and, if $\Gamma_i^j \neq \Gamma_i^{j-1}$ then the new arguments have to be relevant to the previous bargaining context $(\rho_1^{j-1}, \rho_2^{j-1})$.

If $\vec{\Theta}^j$ $(j \geq 0)$ is consistent then the set of agreement-feasible outcomes $AF = \Theta_1^j \cap \Theta_2^j$ is non-empty and an agreement can be selected from AF. If $\vec{\Theta}^j$ $(j \geq 0)$ is inconsistent then the bargaining proceeds to the next round. If in two consecutive rounds of bargaining, the bargaining context is not updated, i.e., for some $j \geq 0, \rho_i^j = \rho_i^{j+1} = \rho_i^{j+2}$ for $i = 1, 2$, then the bargaining reaches a disagreement.

LEMMA 4. *The bargaining protocol defined above terminates.*

Proof: Since the alphabet $\mathcal{P}$ of the language $\mathcal{L}$ (and the alphabet $\mathcal{P}_O \subseteq \mathcal{P}$ of the bargaining outcomes) is finite, there can only a finite number of logically different arguments; i.e., the set $\mathbb{A}_C(\mathcal{L})$ is finite and the set $\mathcal{O}_C$ is also finite. Thus, if the bargaining does not terminate with a disagreement then at some point, both agents will have exhausted the set of arguments and thus will have to increasingly add the members of $\mathcal{O}_C$ in their respective offers and the bargaining terminates with a non-empty set of agreement-feasible outcomes AF. □

Given the negotiation protocol, our example in the preceding section about the vendor agent who argues to convince the buyer to change her position on the price of the house can be described as follows.

EXAMPLE 5. *We will assume that the alphabet $\mathcal{P}$ also contains the following propositional symbols: Y for "*a similar house was sold for $\$y$ last month*", M for "*house prices last month reflects today market*", S for "*houses in the area have become scarce*", D for "*there has been an increase in the demand for houses in the area*", C for "*the market has changed with the price on the up*", and R for "*bargainers should exercise reciprocity*". We'll also assume that $PE \succ_1 P\bar{E} \succ_1 \bar{P}E \succ_1 \bar{P}\bar{E}$ and $\bar{P}E \succ_2 \bar{P}\bar{E} \succ_2 P\bar{E} \succ_2 PE$ are the agents' respective preferences. That is, the buyer's preference on the value of E (whether it should be an early settlement) is conditional on the value of P (whether she has to pay the higher price).*

The negotiation starts off with the initial proposal profile $((\{PE\}, \emptyset), (\{\bar{P}E\}, \emptyset))$. In the next round, the buyer introduces the argument $(\{Y, M, Y \wedge M \Rightarrow \neg P\}, \neg P)$ and the seller introduces the argument $(\{Y, S, D, S \wedge D \Rightarrow C, Y \wedge C \Rightarrow P\}, P)$ to defend their respective positions. In the consecutive round, the seller then defeat the buyer's argument with the argument $(\{C, C \Rightarrow \neg M\}, \neg M)$. This settles the issue on the price of the house with the buyer making a concession and willing to accept any outcome from the set $\{\bar{P}E, \bar{P}\bar{E}, P\bar{E}\}$. However, since the set of agreement-feasible outcome $AF = \{PE\} \cap \{\bar{P}E, \bar{P}\bar{E}, P\bar{E}\}$ remains empty, the buyer then advances the argument $(\{P, R, P \wedge R \Rightarrow \neg E\}, \neg E)$. Since the seller has no counter-argument, he makes a concession and is willing to accept any outcome from the set $\{PE, P\bar{E}\}$. They settle with the outcome $P\bar{E}$.

To formalise the notion of winning argument in an exchange between bargainers, we'll appeal to the argument-based semantics of admissibility, introduced by Dung [4].

DEFINITION 8. *A set Γ of arguments is* **conflict-free** *if there are no two arguments (H, h) and (H', h') such that (H, h) attacks (H', h') or (H', h') attacks (H, h).*

A set Γ of arguments is **admissible** *if it is conflict-free and it defends all of its members against all attackers.*

A set Γ of arguments is **strongly admissible** *if it is admissible and none of the arguments that attack its members belong to an admissible set of arguments.*

The following lemma is trivial since at all stages j of the bargaining, $\mathcal{B}_{\Gamma_i^j} \cup \mathcal{C}$ is required to be consistent.

LEMMA 5. *The sets of arguments contained in the bargaining proposals introduced by the agents according to the bargaining protocol defined above are conflict-free.*

The following axiom requires that bargainers do not ignore strongly admissible sets of arguments that support an agent's position.

SA. If $BP = (\mathcal{C}, \langle B_1, \succeq_1 \rangle, \langle B_2, \succeq_2 \rangle)$ is a bargaining problem and $AG \subseteq \mathcal{O}_\mathcal{C}$ the agreement reached after j rounds of bargaining. If the set of arguments $\Gamma \subseteq \Gamma_i^j$ is strongly admissible then for each outcome $o \in AG$: $\bigwedge_{(H,h)\in\Gamma} h \wedge o$ is consistent.

Intuitively, axiom **SA** requires that, if agent i can present an argument that agent $-i$ can not defeat, then every agreed outcome has to be consistent with the conclusions obtainable from this set of arguments.

We can now state a lemma trivially derived from the definition of strongly admissible sets of arguments.

LEMMA 6. *Given a bargaining problem BP, if a sentence h is supported by a strongly admissible set of arguments Γ from the current bargaining context (ρ_1^j, ρ_2^j) then there does not exist any admissible set of arguments Γ' from the current bargaining context that supports $\neg h$.*

Equipped with Lemma 6 and assuming a belief revision operator $*_{AGM}$ that satisfies the AGM axioms (see [6]), we can now define an argument-augmented bargaining problem of a given bargaining problem.

DEFINITION 9. *Let $BP = (\mathcal{C}, \langle B_1, \succeq_1 \rangle, \langle B_2, \succeq_2 \rangle)$ be a bargaining problem and given a a belief revision operator $*_{AGM}$ that satisfies the AGM axioms. Consider the set of all admissible sets of*

arguments of the base $B_1 \cup B_2$*:* $AS_{BP} = \{\Gamma \subseteq \mathbb{A}_{\mathcal{C}}(B_1 \cup B_2) : \Gamma$ *is strongly admissible* $\}$*. Let* α *denote* $\bigwedge_{(H,h)\in\bigcup_{\Gamma\in AS_{BP}}\Gamma} h$*, then the* **argument-augmented bargaining problem** *of* BP*, denoted by* $\mathcal{A}^{BP}$ *is defined to be* $(\mathcal{C}, \langle B_1^{*_{AGM}}\alpha, \succeq_1\rangle, \langle B_2^{*_{AGM}}\alpha, \succeq_2\rangle)$.

Given a bargaining problem BP, the following bargaining solution can be defined:

$$f^{A\text{-}E}(BP) = \begin{cases} \text{disagreement} & \text{if } AF_{\mathcal{A}^{BP}} = \emptyset \\ f^{E}(\mathcal{A}^{BP}) & \text{otherwise} \end{cases}$$

The following theorem is obvious (given Lemma 6 and the AGM axioms):

THEOREM 2. *The bargaining solution* $f^{A\text{-}E}$ *satisfies axiom* **SA**.

Given the negotiation protocol above and our proposed argument-based bargaining framework, we are interested in finding the equilibrium strategies in a bargaining situation. Note that, in a strategic model of bargaining, a bargainer's beliefs and preferences are her private information and the bargaining progresses when the agents exchange their proposals, in our bargaining protocol, by simultaneously putting them on the negotiation table. However, in the presence of incomplete information, the agents clearly have the incentives not to reveal their true preferences and beliefs. Therefore, to develop a tractable strategic model of bargaining, we'll need to make a number of assumptions including an enforceable penalty mechanism which is also underlying the negotiation framework developed by Rosenschein and Zlotkin [13, 17]. Given such assumptions, we have developed a symmetric Nash equilibrium strategy for the bargainers based on the minimal concession strategy introduced by Dung et al. [5]. This consideration is beyond the scope of the present paper and will be included in our future work.

5. CONCLUSION AND FUTURE WORK

In this paper we introduced an axiomatic model of bargaining with logical beliefs and goals for the purpose of bargaining analysis. To the best of our knowledge, our model is the first logic-based axiomatic model of bargaining that does not suffer the problem of syntax-sensitivity while still ensuring that our proposed bargaining solution is uniquely characterized by a concise set of intuitive axioms (see e.g., [16, 15]). This is the most important contribution of our paper. Moreover, our framework allows for a separation between the bargainers' beliefs and their respective goals. This is important because not all beliefs of the agents that are relevant to the negotiation will necessarily end up on the negotiation table. Many of them may be used only for the agents to make decision about whether to accept an offer or what counter-offer to be made.

We have also taken into account the problem of dynamic negotiation in which the bargainers' bottom lines could be changed during the negotiation. The problem is challenging, particularly in the context of incomplete information. We appeal to the formalism of argumentation framework to allow for the accommodation of new and revised beliefs and their effects on the bargainers' bottom lines. We are currently investigating a strategic model of bargaining to complement our axiomatic model. This is also the final step in realising the famous Nash program. This work will be reported in our future papers.

From a multi-agent systems point of view, it is always an important question to investigate the computational complexity of the solutions and the concepts we have proposed. This also remains a challenge to be addressed in the future work.

Acknowledgements

This work was supported by the Australian Research Council (ARC) grants DP0987380 and DP110103671. The authors would like to thank Prof Ryszard Kowalczyk for his support. The authors would also like to thank the four anonymous reviewers of AAMAS-2012 for their very detailed comments and suggestions.

6. REFERENCES

[1] L. Amgoud and C. Cayrol. Inferring from inconsistency in preference-based argumentation frameworks. *J. Autom. Reason.*, 29(2):125–169, 2002.

[2] K. Binmore. *Game Theory and the Social Contract, Vol. 2: Just Playing*. The MIT Press, Mar. 1998.

[3] R. Booth. Social contraction and belief negotiation. *Information Fusion*, 7(1):19–34, 2006.

[4] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence Journal*, 77:321–357, 1995.

[5] P. M. Dung, P. M. Thang, and F. Toni. Towards argumentation-based contract negotiation. In *Procs. of Computational Models of Argument: COMMA*, pages 134–146, 2008.

[6] P. Gärdenfors. *Knowledge in Flux: Modeling the Dynamics of Epistemic States*. The MIT Press, 1988.

[7] S. Konieczny, J. Lang, and P. Marquis. DA2 merging operators. *Artificial Intelligence*, 157(1-2):49 – 79, 2004.

[8] T. Meyer, N. Foo, R. Kwok, and D. Zhang. Logical foundations of negotiation: outcome, concession and adaptation. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI-04)*, pages 293–298, 2004.

[9] T. Meyer, N. Foo, R. Kwok, and D. Zhang. Logical foundations of negotiation: strategies and preferences. In *Proceedings of the 9th International Conference on the Principles of Knowledge Representation and Reasoning (KR'04)*, pages 311–318, 2004.

[10] J. Nash. The bargaining problem. *Econometrica*, 18(2):155–162, 1950.

[11] J. Nash. Two-person cooperative games. *Econometrica*, 21(1):129–140, 1953.

[12] M. J. Osborne and A. Rubinstein. *Bargaining and Markets*. Academic Press, 1990.

[13] J. S. Rosenschein and G. Zlotkin. *Rules of Encounter: Designing Conventions for Automated Negotiation Among Computers*. MIT Press, Cambridge, Massachusetts, 1994.

[14] M. Shubik. *Game Theory in the Social Sciences: Concepts and Solutions*. MIT Press, Cambridge, 1982.

[15] D. Zhang. A logic-based axiomatic model of bargaining. *Artif. Intell.*, 174(16-17):1307–1322, 2010.

[16] D. Zhang and Y. Zhang. An ordinal bargaining solution with fixed-point property. *J. Artif. Int. Res.*, 33:433–464, November 2008.

[17] G. Zlotkin and J. S. Rosenschein. Mechanisms for automated negotiation in state oriented domains. *Journal of Artificial Intelligence Research*, 5:163–238, 1996.

Session 5C
Emergence

Crowd IQ - Aggregating Opinions to Boost Performance

Yoram Bachrach
Microsoft Research
yobach@microsoft.com

Thore Graepel
Microsoft Research
thoreg@microsoft.com

Gjergji Kasneci
Microsoft Research
gkasneci@gmail.com

Michal Kosinski
University of Cambridge
mk583@cam.ac.uk

Jurgen Van Gael
Microsoft Research
jurgen.vangael@gmail.com

ABSTRACT

We show how the quality of decisions based on the aggregated opinions of the crowd can be conveniently studied using a sample of individual responses to a standard IQ questionnaire. We aggregated the responses to the IQ questionnaire using simple majority voting and a machine learning approach based on a probabilistic graphical model. The score for the aggregated questionnaire, Crowd IQ, serves as a quality measure of decisions based on aggregating opinions, which also allows quantifying individual and crowd performance on the same scale.

We show that Crowd IQ grows quickly with the size of the crowd but saturates, and that for small homogeneous crowds the Crowd IQ significantly exceeds the IQ of even their most intelligent member. We investigate alternative ways of aggregating the responses and the impact of the aggregation method on the resulting Crowd IQ. We also discuss Contextual IQ, a method of quantifying the individual participant's contribution to the Crowd IQ based on the Shapley value from cooperative game theory.

Categories and Subject Descriptors

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

General Terms

Algorithms, Economics

Keywords

Human Intelligence, Opinion Aggregation, Shapley Value

1. INTRODUCTION

Human intelligence and group decision processes have been extensively studied for many years. However, in recent years internet-based technologies have dramatically changed the ways in which people interact, socialize and communicate. People exchange information through online social networks, communicate their opinions through websites and rely on internet sources in their economic decisions. The accessibility of such information makes it easier to *aggregate* the opinions of many individuals and examine the quality of decisions based on such aggregated information.

While *collecting* the opinions of individuals is easy, it is more difficult to decide how to *aggregate* the opinions in order to *reach decisions*, and *measure the quality* of the decisions based on the aggregated information. There is often a considerable variance in the individual opinions so reaching optimal decisions is not a trivial task. Moreover, once a decision has been reached, it is impossible to compare its quality to other possible decisions that have not been taken. Consequently, although aggregating the opinions of many individuals is appealing, performing the aggregation process successfully requires answering many important questions: How does the quality of the decisions depend on the group size? Is it better to base decisions on the opinions of the best individual in a group, or is it better to rely on other people's opinions as well? How can one measure the contribution of individuals to the quality of the decisions? Do individual contributions depend more on the individual's skill or on her similarity to the group?

In an attempt to answer those questions we explore the process of aggregating participants' responses to IQ items on the established IQ test[1] - Raven's Standard Progressive Matrices (SPM) [26]. We treat participant' responses to an IQ test, expressed independently in a setting similar to popular crowdsourcing environments such as Amazon Mechanical Turk, as their opinions regarding the correct solution. We aggregate those individual opinions using majority vote or a machine learning aggregator to reach a decision regarding a correct response to each of the items. We then score this test solved by the crowd using standard scoring procedures, referring to the resulting IQ score as *the Crowd IQ*.

People's responses to the SPM IQ test offer a convenient and robust environment to study the aggregation of individual opinions. First, SPM offers a set of non-trivial problems engaging a range of human cognitive abilities with well-defined correct response and limited number of possible solutions. Second, an individual's IQ score is an elegant measure of one's mental abilities and is a good predictor of behavior

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

[1]We refer to individuals who have completed an IQ test as *participants*, questions in such a test as *items*, and to participants' answers as *responses*.

and performance in a broad spectrum of contexts including job and academic performance, creativity, health-related behaviors and social outcomes [10, 11, 18, 28]. Third, the Crowd IQ score provides a convenient quality measure of the crowd's aggregated decision. Finally, IQ scores provide a uniform performance metric that allows exploring the relationship between individual and crowd performance.

Our Contribution: We examine the properties of the Crowd IQ and show that aggregating opinions of crowd members can significantly boost the expected quality of the decision. We show that the Crowd IQ grows quickly with its size but then saturates, indicating diminishing returns from each additional member. We also show that for homogeneous crowds the Crowd IQ significantly exceeds the IQ of the most intelligent member in the crowd. Finally, we show that an individual's contribution to the Crowd IQ is not solely related to the participant's IQ but also depends on the uniqueness of her contribution in the context of a given crowd.

2. RELATED WORK

Many papers deal with aggregating the opinions of multiple agents to reach high quality decisions. Social choice theory deals with joint decision making by self-interested agents (see [29] for a broad discussion of this field), and is a key research area in artificial intelligence and multiagent systems. The Condorcet Jury Theorem from social choice theorey provides theoretical bounds regarding the probability of a set of agents to reach the correct decision under majority voting [1, 20, 15]. However, the Condorcet Jury Thereom uses strong assumptions which may not hold in practice, such as requiring votes to be completely independent. Our study can be viewed as an empirical examination of this topic using data from IQ questionnaires.

Another related field is judgment aggregation [16] which deals with aggregating group members' individual judgments on some interconnected propositions, expressed in a formal logic language, into corresponding collective judgments on these propositions. These fields have also been examined by computer scientists, who found practical and computationally tractable ways of performing such aggregation, ranging from machine learning approaches [12] to prediction markets [22]. However, the IQ test items are not interconnected, and our focus is on quantifying and decomposing the group's performance. In our paper we also ignore the complications of aggregating agent opinions when such agents are self-interested and may wish to influence the aggregated choice [9, 8].

Human intelligence has been a central topic in psychology. Psychologists noted that people's performance on many cognitive tasks is strongly correlated, leading to the emergence of a single statistical factor, typically called "general intelligence" [32, 10, 18, 28]. Recent work extends this to "collective intelligence" for performance of *groups* of people in joint tasks [35], which is not strongly correlated with the maximal or average intelligence of the group members. However, this approach examines explicit collaboration and interaction between the group members, where the social interaction may sometimes even hinder performance [17], whereas we focus on information aggregation. Approaches more similar to ours are [7, 19] and [34] which even proposes a machine learning aggregator for image labeling in a crowdsourcing environment. However, our focus is on the impact of the aggregation methods and methods for quantifying individual contribution based on a standardized IQ test.

3. METHODOLOGY

We now describe the IQ test we used, our dataset of participants' responses, and the aggregation methods used to establish the crowd's solution to the test.

3.1 Standard Raven Progressive Matrices Test

The IQ test used in this study, Raven's Standard Progressive Matrices [23, 25], was developed by John C. Raven [24]. It is a multiple choice non-verbal intelligence test drawing on Spearman's theory of general ability [32] and consists of $m = 60$ matrices with one element missing and $k = 8$ possible responses. Matrices are separated into five sets of 12 and within each set the problems become increasingly difficult. A sample item, similar[2] to those used in the SPM is shown on Figure 1.

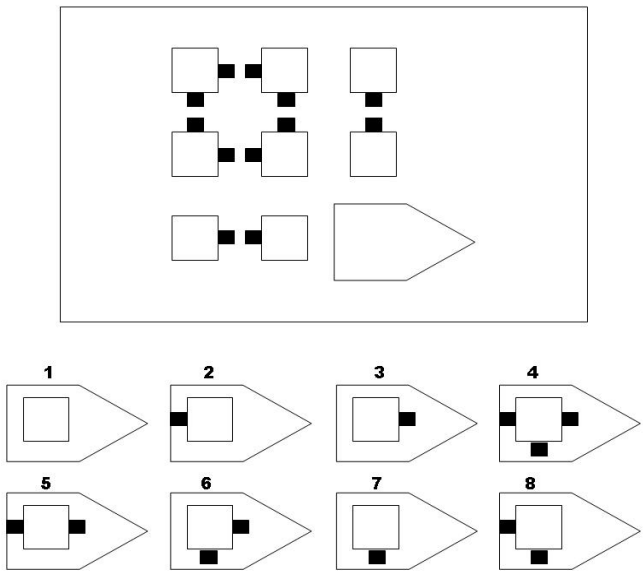


Figure 1: Item similar to those in the SPM test

Raven's SPM and its other forms (Advanced and Colored Progressive Matrices) are one of the most popular intelligence tests used in both research and clinical settings, as well as in high-stake contexts such as in military personnel selection and court cases [25].

3.2 Dataset and Scoring

Our sample consisted of 138 individuals, aged 15-17, who filled the SPM during its standardization for the British market in the year 2006 [23]. The sample is representative of the British population.

The standard scoring procedure described in the test's manual was used to calculate individual and Crowd IQ scores [23]. The manual provides tables for translating the number of

[2]The SPM test is copyright protected, so we can only provide an item similar to those in the actual test, rather than a sample item from the test itself.

correct responses (raw score) into an IQ score. The IQ scale characteristic for SPM (and most other intelligence tests) is standardized on a representative population to have a normal distribution with an average score of 100 and standard deviation of 15. Hence, IQ scores allow for convenient comparisons between individuals, and comparing individual performance with the general population. The distribution of the raw scores and the IQ scores in our sample is shown on Figures 2 and 3. The average number of correct responses in the dataset is 36.04, with a standard deviation of 5.49. The average IQ score is 99.57 with a standard deviation of 14.16.

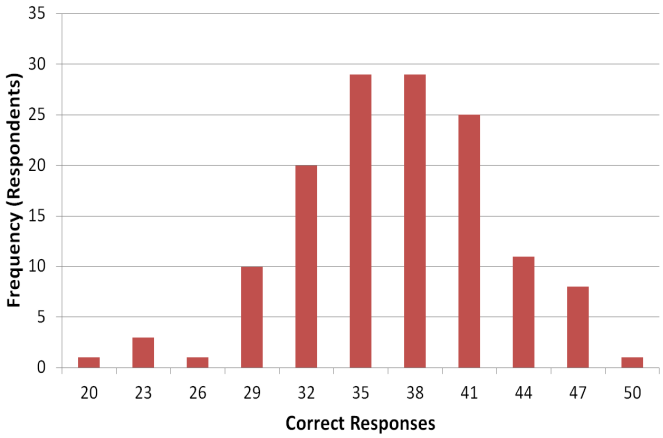


Figure 2: Histogram of raw IQ scores

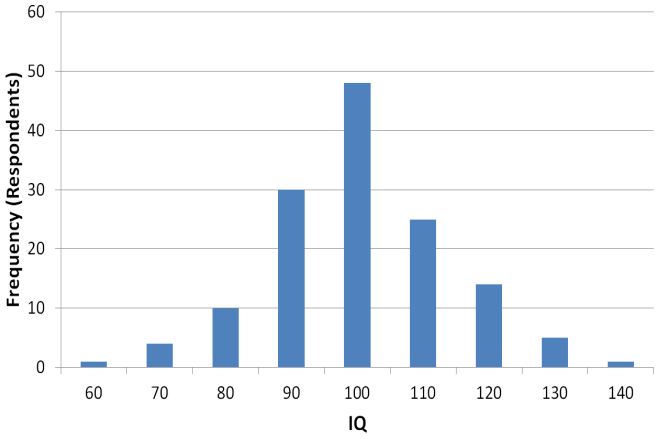


Figure 3: Histogram of IQ scores

3.3 Aggregating Individual Responses

Consider an IQ questionnaire consisting of m items, each a multiple choice item with k possible responses. Denote the possible responses $K = \{1, \ldots, k\}$. The questionnaire is administered to a set N of n participants, each providing a response for each of the items. Let $r_j^i \in K$ be the response provided by participant $j \in N$ to item i and r_j be the responses provided by participant $j \in N$ to all items, so $r_j = (r_j^1, r_j^2, \ldots, r_j^m)$. We call r_j the *filled questionnaire* for participant j.

An aggregation method f takes the filled questionnaires of the participants, $r_1, \ldots, r_n$, and outputs a single filled questionnaire agg_N^f, which contains a response to each of the items, so $agg_N^f = (r_{agg}^1, \ldots, r_{agg}^1)$ where $r_{agg}^i \in K$ is the response chosen to the item i. The aggregated questionnaire agg_N^f is scored using the standard scoring key.

We now briefly describe the two aggregators used in this paper: *A simple majority aggregator with lexicographical tie-breaking* (**MAJ**), and *a machine learning aggregator* (**ML**).

3.3.1 Simple Majority Aggregation

The MAJ aggregator considers each item of the IQ questionnaire separately. It chooses the most common response as the "correct" response, and thus bases the decision on the choice made by the majority of the participants. If two or more responses are selected an equal number of times (tie) the first one in the lexicographical order is selected.

The MAJ aggregator has several limitations. First, it does not use the information obtained from the responses to one of the items to decide how to aggregate the responses to another item. For example, if a user u has answered all items correctly until item i and user v has answered all items incorrectly until item i, when aggregating the responses to an item $i + 1$, it might be desirable to give u's opinion more weight than v's opinion. Further, the MAJ aggregator makes no assumptions about the data-generating process other than that the correct response should be chosen more frequently than any of the incorrect ones.

3.3.2 Machine Learning Based Aggregation

Our ML aggregator addresses the MAJ's limitations. Similarly to the MAJ aggregator, the goal of the ML one is to take questionnaires completed by several participants and output a single questionnaire with inferred correct responses. In contrast to the MAJ aggregator, this is a non-simple aggregation, in which the inferred response to an item also depends on responses provided to other items. The model attempts to make better inferences about the correct responses to items by jointly modeling the participants' aptitude and the correct responses. The underlying assumption is that each participant has an associated probability of knowing the correct response to an item, their aptitude, and that they will randomly guess the answer if they do not know the correct response. The ML aggregator designed for this study employs probabilistic graphical models [21, 13].

Probabilistic Graphical Models allow structurally describing the generative process assumed to underlie the observed data in terms of latent and observed random variables. In the context of Crowd IQ, information like the correct response to an item or the intelligence of a participant would be modeled as unknown latent variables whereas the given response to an item by a user would be an observed variable. The structure of the model is then determined by the conditional independence assumptions made about the variables in the model. Pearl [21] introduced Bayesian Networks to encode assumptions of conditional independence in the form of a graph whose nodes represent the variables and whose edges describe the dependencies between variables. We use the more general notion of a factor graph, see e.g. [13], to describe the factorial structure of the assumed joint probability distribution among the variables. Once the structure of the model is defined in terms of a factor graph, observed variables can be set to their observed

values. Then approximate message passing algorithms [13] can infer marginal probability distributions of unknown variables of interest such as the correct response to an item or the intelligence of a participant.

Graphical Model for IQ Response Data: We wish to infer the correct responses, so the graphical model contains a set of random variables $y_q \in Y_q$ that represent the correct response to each of the items q. Each y_q takes discrete values in the set Y_q of possible responses q. The model's initial ignorance about the correct response is expressed by assuming a uniform prior distribution over responses, $y_q \sim$ Uniform. We also wish to take into account the (unknown) aptitude of participants in order to weigh their responses appropriately. The aptitude of each participant $i \in N$ is represented by a random variable $g_i \in R$, which can be interpreted as the probability that the participant would know the correct response. We choose uniform prior densities for these variables, $g_i \sim \text{Beta}(1.0, 1.0)$. Here, Beta represents the family of beta distribution, which allows us to compactly represent (unimodal) beliefs over the g_i. We also introduce a uniform "guessing" distribution $B =$ Uniform, which models the choice of response when the participant is assumed to be guessing. Participant i's response r_j to item q, is assumed to be drawn from the following distribution:

$$r_j \begin{cases} \sim B & \text{with probability} \quad (1 - g_i) \\ = y_q & \text{with probability} \quad g_i \end{cases}$$

This means that with probability g_i participant i chooses the correct response y_q and with probability $1 - g_i$ she randomly guesses the answer based on the guessing distribution B. Figure 4 illustrates this probabilistic graphical model in the form of a factor graph. Note that the grey boxes represent plates which indicate repetition of the contained substructure of the graphical model. In this case, q ranges over the available items, i ranges over the available participants, and j ranges over the available responses.

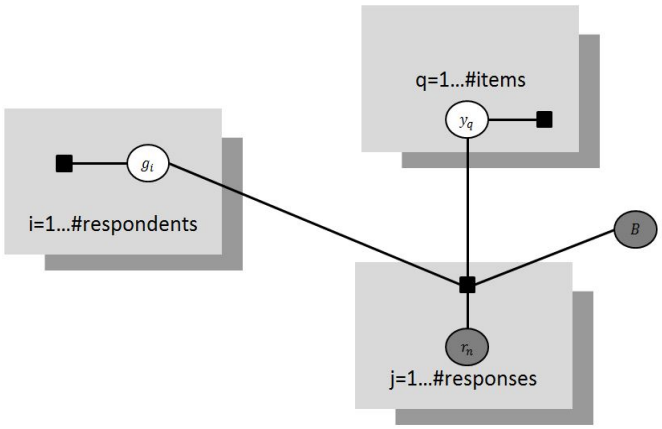


Figure 4: Factor graph for the ML aggregator

Inference in the model is performed using approximate message passing (see [13] for details)[3]. As a result we obtain a discrete marginal posterior distribution over responses to each of the items, representing the model's belief about the correct response in light of the observed data. As a by-product we also obtain the posterior marginal densities over the aptitude variables g_i for each user. To minimize the probability of error we choose the response with the maximum posterior for each item as the aggregated response.

[3]Our implementation used the Infer.net library. For details regarding Infer.net see: http://research.microsoft.com/en-us/um/cambridge/projects/infernet/).

3.4 Contextual IQ: Individual's Contribution to the Crowd IQ

We now discuss our approach for quantifying an individual's contribution to a Crowd IQ. Intuitively, individuals of high IQ are likely to contribute more towards the aggregate IQ of the crowd, i.e. the individual's IQ divided by the sum of the IQ scores of all the members of the crowd. However, as individuals' skills and knowledge may differ, the individual contribution depends also on the relationship between the patterns of her responses and those of the other members of the crowd (or context). For example, imagine a crowd that can correctly solve a subset of questions A but is unable to provide a correct answer to questions B. Adding another individual to this crowd that can correctly solve questions A but does not know correct responses to questions B would not increase the Crowd IQ score, while adding an agent that knows correct responses to questions B can potentially boost Crowd's performance. We refer to this relative boost as a *Contextual IQ*. Our approach to quantifying contextual IQ is based on the Shapley value [30], a concept from cooperative game theory.

3.4.1 Measuring Impact on Performance Using the Shapley Value

Cooperative game theory studies the behavior of selfish agents who must cooperate to achieve a goal, and analyzes how the rewards from such cooperation should be distributed among the agents. Solution concepts from game theory can be used to find reward distributions fulfilling desirable properties, such as being fair or stable. Our methodology examines the game where the agents are the participants filling the IQ test, and where the value of a coalition of agents is the Crowd IQ of that coalition.

The Shapley value [30] can be viewed as a "power index", a tool for measuring an individual's *contribution* or *importance* in the success of a team of agents, or for quantifying an agent's ability to influence a game's outcome [31, 6]. The Shapley value was used for measuring political influence of parties forming a coalition in legislative bodies [14], analyzing network reliabilitiy [5, 2, 4] and fair cost allocation [27, 33]. Further, the Shapley value is the only imputation fulfilling certain fairness axioms [30].

The Shapley value relies on the marginal contribution of an individual — the amount of additional utility gained when that individual joins the crowd. We denote by $\pi \in \mathcal{S}_n$ a permutation of the agents, so $\pi : \{1, \ldots, n\} \to \{1, \ldots, n\}$ and π is onto. Denote by Γ_i^π the predecessors of i in π, so $\Gamma_i^\pi = \{j | \pi(j) < \pi(i)\}$. Agent i's marginal contribution in the permutation π is $m_i^\pi = v(\Gamma_i^\pi \cup \{i\}) - v(\Gamma_i^\pi)$. The Shapley value of an individual is her marginal contribution averaged across all possible permutations of the individuals.

Definition 1. *The Shapley value is the imputation* $(\phi_1(v), \ldots, \phi_n(v))$ *where*

$$\phi_i(v) = \frac{1}{n!} \sum_{\pi \in \mathcal{S}_n} m_i^\pi = \frac{1}{n!} \sum_{\pi \in \mathcal{S}_n} \left(v\left(\Gamma_i^\pi \cup \{i\}\right) - v\left(\Gamma_i^\pi\right) \right)$$

Consider a set N of agents (participants) filling an IQ questionnaire, with m items and a set K of k possible responses to each item, and a set $C \subseteq N$ to be used as a crowd (coalition). Denote the responses of participant $i \in C$ as $r_i \in K^m$, and the set of responses of all the agents in C as $r_C = (r_1, \ldots, r_{|C|})$. Thus the space of possible responses of each agent is $A = K^m$, and the responses of *all* the participants are in the space $A^{|C|}$. Consider an aggregator $f : A^{|C|} \rightarrow A$ which maps the responses of all agents to a single filled questionnaire.

As in Section 3.3, we denote the filled questionnaire obtained by applying the aggregator f to the responses of the agents in C as agg^f_C. In Section 3.3 we defined the aggregate IQ of a crowd C as the IQ score of the filled questionnaire agg^f_C. We define a cooperative game v_f that maps any subset $C \subseteq N$ of agents into their aggregate Crowd IQ (the IQ score of agg^f_C, the aggregate response for the crowd C). This cooperative game over the set N of agents is defined with the following characteristic function: $v_f(C) = IQ(agg^f_C)$, and is called the *Aggregate IQ Game*.

In the Aggregate IQ Game, the "reward" of any coalition C is the aggregate IQ of the crowd C, and $v_f(N)$ is the aggregate IQ of the entire agent set N. Our goal is to decompose $v_f(N) = IQ(agg^f_N)$, the total aggregate IQ score obtained by the grand coalition N of all agents, to the individual contribution of each agent. We refer to the set N of all agents as the *context* in which we measure an agent's individual contribution. We are thus seeking a vector $\vec{p} = (p_1, \ldots, p_n)$ such that $\sum_{i=1}^{n} p_i = v_f(N)$ where p_i reflects i's fair contribution to the total IQ score. Due to the properties of the Shapley value we can use it to fairly decompose the Crowd IQ score. We define agent i's *Contextual IQ* (for the given context N) as its Shapley value in the above Aggregate IQ game. One interpretation of this definition is that the aggregate IQ of the crowd is decomposed into the contribution, in IQ points, of each participant. These contextual IQ scores sum up to the total aggregate IQ of the crowd N, and a participant has a higher contextual IQ than another participant if she is expected to have a higher positive influence on the aggregate Crowd IQ score of a subset of participants selected at random from the entire crowd N.

By Definition 1, the contextual IQ is the expected increase in Crowd IQ when adding i to her predecessors in a random permutation of the agent set N. Note that m^π_i is the increase in Crowd IQ when adding i to a specific agent subset, Γ^π_i, and the Shapley value is the *average* of these increments in Crowd IQ across all agent permutations. Obviously, an individual's contextual IQ (Shapley value) is strongly affected by her IQ score, as responding correctly to more items increases the marginal contribution for m^π_i for many permutations π (assuming a reasonable aggregator).

Computing contextual IQ using formula 1 requires a running time exponential in the number of the agents. We used the approach of [3] for computing the Shapley value, which offers a very high accuracy and a tractable polynomial running time. This algorithm samples many agent subsets (or more precisely permutations) of the crowd and averages the marginal contribution of the target agent in them to obtain an accurate approximation of the Shapley value.

4. CROWD SIZE AND CROWD IQ

First, we unveil the relationship between the Crowd IQ and its size. Figure 5 shows the relationship between the size of the crowd (number of participants) and its IQ established using both MAJ and ML aggregators as discussed in Section 3.3. Each point in the plot is the average Crowd IQ for $q = 300$ randomly selected crowds of the specified size. Such repetitive sampling minimizes the influence of the selection bias on the Crowd IQ estimates.

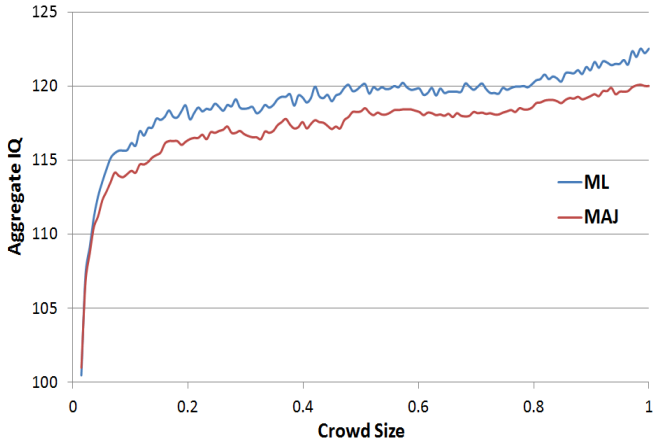


Figure 5: Crowd IQ scores based on the MAJ and ML aggregators for different crowd sizes

Figure 5 shows that the Crowd IQ quickly increases with the crowd size but saturates after reaching the crowd size of 14 participants and IQ of about 115, roughly one standard deviation increase above the population mean. These results indicate that the quality of the crowd decision is significantly higher than the average IQ of its members. However, returns from increasing the crowd size rapidly diminish after a certain size is reached. Also, Figure 5 shows that a machine learning based aggregation consistently outperforms simple majority aggregation, by learning which users provide correct responses more reliably.

5. SMARTER THAN A CROWD?

Here we investigate whether it is better to base decisions solely on the opinions of the high-performing individuals in a group, or to rely on other peoples' opinions as well. One way of examining this is to determine whether the Crowd IQ is likely to exceed the IQ of the smartest individual in the crowd. We use the approach described in Section 4 to compute the relationship between Crowd IQ and its size and we plot the maximal individual IQ for any given crowd size.

Figure 6 shows the relation between the crowd size, expected Crowd IQ, and expected maximal IQ for the entire dataset used in this study. It is clear that in large crowds characterized by a wide distribution of IQ scores, the maximal IQ consistently exceeds Crowd IQ. While Crowd IQ for this population saturates around 115-120 IQ points, the chance of the crowd encompassing individuals with extreme IQ scores increases with the sample size.

However, it is common for the crowds to be composed of individuals characterized by the similar IQ (*homogeneous* crowds). For instance, the IQ of students of advanced degrees is likely to be homogenous and relatively high,as IQ is correlated with academic performance. A homogeneous crowd is less likely to contain an individual with an IQ score much higher than the average IQ score in the crowd, so

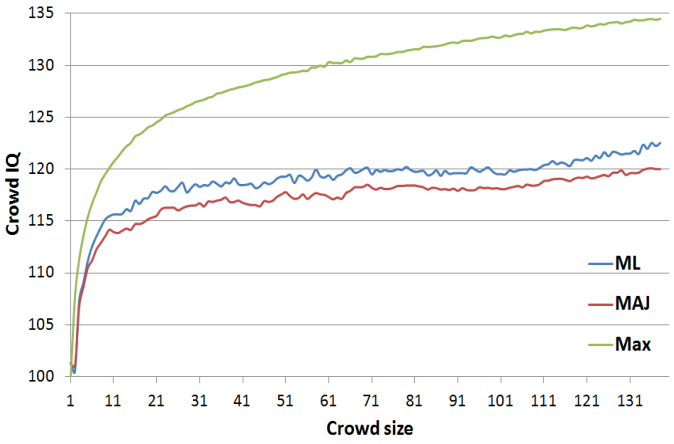


Figure 6: Crowd IQ and maximal IQ (entire dataset)

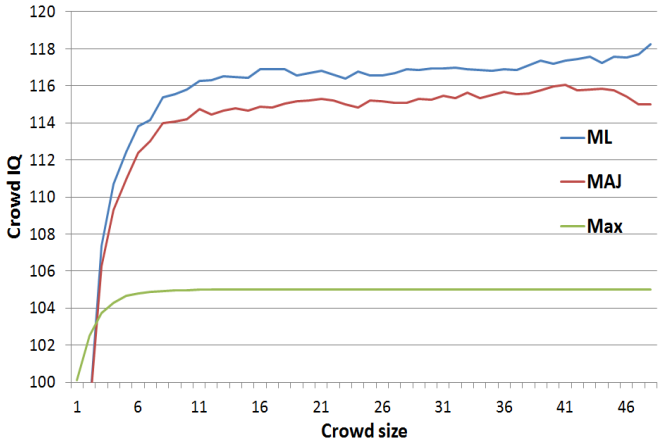


Figure 7: Crowd IQ and maximal IQ for $P_{[95,105]}$

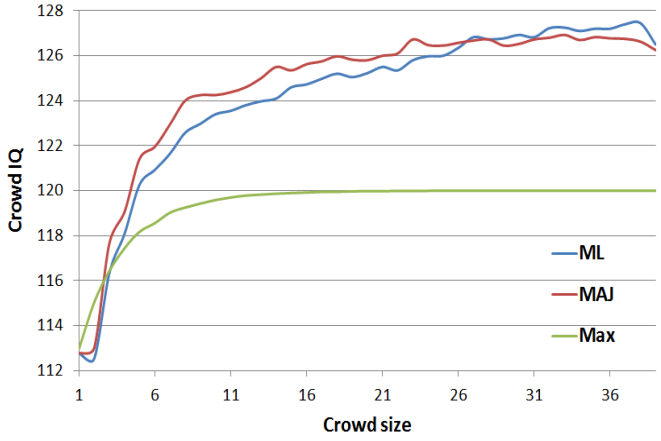


Figure 8: Crowd IQ and maximal IQ for $P_{[110,120]}$

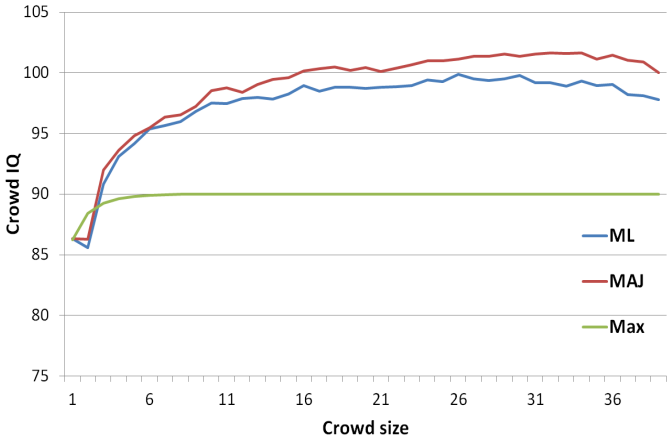


Figure 9: Crowd IQ and maximal IQ for $P_{[80,90]}$

the performance of the crowd may be superior to that of its smartest individual. To examine this issue, we split our sample into a set of homogeneous subsamples based on the individual IQ scores. Subsamples are denoted by $P_{[L,H]}$, where $[L, H]$ represents the range of participants' IQ scores. Thus, subsample $P_{[110,120]}$ contains individuals with IQ scores between 110 and 120.

Figures 7, 8, and 9 for subsamples $P_{[95,105]}$, $P_{[110,120]}$, and $P_{[80,90]}$[4] show that the Crowd IQ greatly exceeds its most intelligent member's IQ in homogeneous crowds. Also, a homogeneous crowd's advantage over its smartest member increases as it grows.

Interestingly, the simple MAJ aggregator outperforms the ML aggregator's in homogeneous high and low IQ populations ($P_{[110,120]}$ and $P_{[80,90]}$). A possible explanation of this phenomenon might be related to the lack of outstanding individuals in such crowds, that could be used by ML aggregator to boost its performance. However, this clearly does not apply to the similarly homogeneous $P_{[95,105]}$ subsample where ML outperforms MAJ aggregator. Further, Figure 9 shows the decrease in performance of both aggregators for very big crowd sizes. Aggregated performance might be affected by especially popular but incorrect responses to the difficult IQ items that may, for larger crowds, suppress the correct but unpopular responses.

These results indicate that decisions based on the aggregate opinions of rather homogeneous crowds are of a higher quality than those based solely on the opinion of their most intelligent member. On the contrary, in populations characterized by a wide range of individual performance levels, smartest members outperform the crowd. Note, however, that in all cases the Crowd IQ greatly exceeds the IQ of the average member of the crowd, as discussed in Section 4.

6. INDIVIDUAL IQ AND CONTEXTUAL IQ

We now focus on the relationship between individual and contextual IQ. A participant's contextual IQ is the expected increase in Crowd IQ from adding that participant to a random permutation of the crowd's members. Given the opportunity to add another member to a team of an unknown composition, the optimal choice is the agent with the highest contextual IQ. We now discuss the correlation between individual IQ and contextual IQ using the crowd composed of the entire population of $n = 138$ participants. Figure 10 presents a scatter plot correlating the participants' IQ scores

[4]The number of participants in these subsamples are: $|P_{[95,105]}| = 48$, $|P_{[110,120]}| = 39$, $|P_{[80,90]}| = 39$.

with their contextual IQ scores.

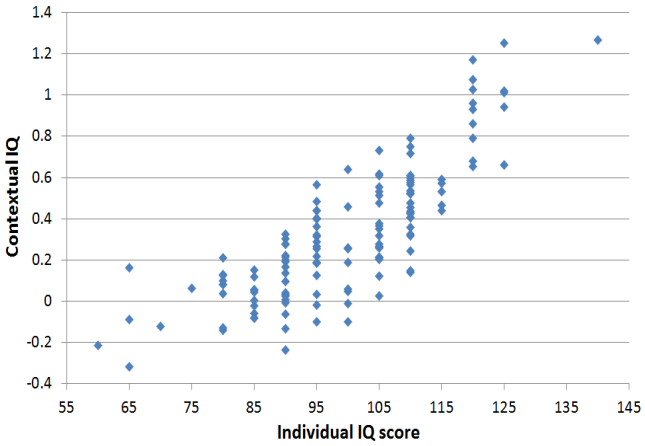


Figure 10: IQ and Contextual IQ

As Figure 10 shows, there is a positive correlation between IQ and Contextual IQ, but also a high variance of contextual IQs for participants of equal IQ. For example, for the above-average IQ of 105, contextual IQ ranges from very high, through negligible to negative. Thus, even if more intelligent people are generally contributing more to the Crowd IQ, the value of their contribution varies and may even be negative. This indicates that although adding the participant with highest IQ score is a good heuristic, better results can be achieved by using the Contextual IQ approach.

A participant's contextual IQ depends on the aggregated IQ, which in turn depends on the aggregator used. The cooperative game used to generate Figure 10 was based on the MAJ aggregator. Measuring the Crowd IQ under a different aggregator (e.g. the ML aggregator), changes the contextual IQ scores of the participants. For example, as shown in Figure 5, the Crowd IQ of all the participants is slightly higher under the ML aggregator, and as the contextual IQ scores must sum up to the total Crowd IQ, the sum of the contextual IQ scores under the ML aggregator would be slightly higher than their sum under the MAJ aggregator.

We now examine the extent to which a participant's contextual IQ is sensitive to the aggregator. Figure 11 shows a plot correlating a participant's Contextual IQ under the MAJ aggregator and her Contextual IQ under the ML aggregator.

Figure 11 shows a high correlation between participant's contextual IQ under the MAJ and ML aggregators (correlation coefficient of over 0.95). Thus, although the aggregator has a slight impact on contextual IQ, the key factors affecting contextual IQ are the participant's IQ and the participant match with the crowd (i.e. the uniqueness of her contribution).

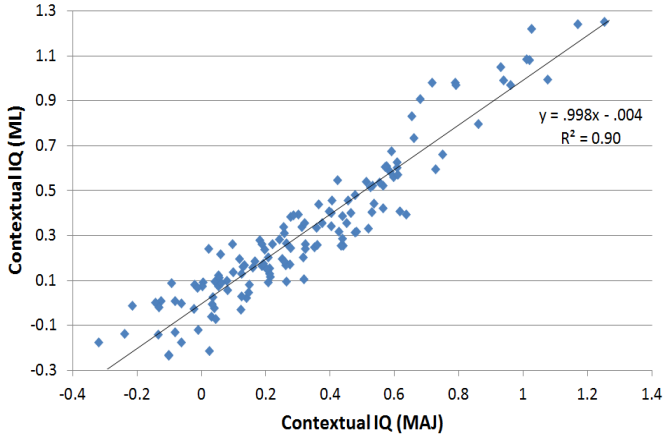


Figure 11: Contextual IQ under the majority and machine learning aggregator

7. CONCLUSIONS AND LIMITATIONS

In this paper we focused on measuring the quality of decisions based on aggregated opinions of the crowd. We proposed that the aggregation of crowd opinions can be conveniently studied using the samples of individual responses to standardized ability tests, such as Raven's Standard Progressive Matrices. One of the main advantages of such samples is the ability to quantify both individual and crowd performance on the same scale.

We showed that decisions based on the aggregated opinions of the crowd are of higher quality than the average quality of the individual member's opinions. A crowd of 14 individuals has an average IQ score of around 115, one standard deviation above the average individual score. This finding is especially important for crowdsourcing environments where it is hard or impossible to detect highly performing individuals prior to the decision making process. We showed that the decisions based on the aggregated opinions of homogeneous crowds are better than the decisions based on the crowds' best performing members, whereas the best approach for a heterogeneous population is to identify the best performing individual and base the decision on her opinions. Our findings indicate that while an individual expert can be smarter than the general opinion pool, she cannot compete against the crowd of her highly performing colleagues, even if she outsmarts each of them individually.

Finally, we proposed the concept of contextual IQ that allows measuring individual contributions towards the aggregate IQ of the crowd. We showed that although the contribution is typically higher when the individual's IQ is higher, it also depends on the uniqueness of individual's contribution in the context of a given crowd.

Limitations: Our approach has several limitations. First, in our setting the crowd members expressed their opinions independently. Such a situation is typical for many crowdsourcing environments, but our findings may not be relevant to contexts in which crowd members can discuss or compare their opinions. Second, we did not collect our data in an actual crowdsourcing environment, where the structure of the individual's opinion might be different from what we observed in our sample. For instance, while the dominant strategy for filling the SPM IQ test is to attempt to answer the item even if the correct response is unknown to the individual, in some crowdsourcing environments (e.g. Amazon Mechanical Turk) individuals may be punished for providing incorrect responses, and thus usually refrain from doing so. Finally, the performance of the ML model was not significantly or consistently higher than of the simple MAJ aggregator which suggests that there is a field for improvement. For example, a more advanced model could allow for

non-uniform distribution of incorrect responses.

Many questions are open for future research. Are there better aggregators that give a stronger boost to Crowd IQ? Which aggregators are better fitted for large and small crowds? Do such aggregation effects also occur in domains other than IQ and real-life crowdsourcing settings? Specifically, would aggregating responses in crowdsourcing settings, such as Amazon's Mechanical Turk, yield similar results? Can the match between an individual and a crowd be predicted using features such as personality, gender or country of origin? Can contextual IQ be efficiently used to select small crowds that would have a high performance in real-world tasks?

8. REFERENCES

[1] D. Austen-Smith and J.S. Banks. Information aggregation, rationality, and the condorcet jury theorem. *American Political Science Review*, pages 34–45, 1996.

[2] H. Aziz, O. Lachish, M. Paterson, and R. Savani. Power indices in spanning connectivity games. *Algorithmic Aspects in Information and Management*, pages 55–67, 2009.

[3] Y. Bachrach, E. Markakis, E. Resnick, A.D. Procaccia, J.S. Rosenschein, and A. Saberi. Approximating power indices: theoretical and empirical analysis. *Autonomous Agents and Multiagent Systems*, 2010.

[4] Y. Bachrach and J.S. Rosenschein. Power in threshold network flow games. *Autonomous Agents and Multi-Agent Systems*, 18(1):106–132, 2009.

[5] Y. Bachrach, J.S. Rosenschein, and E. Porat. Power and stability in connectivity games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 2*, pages 999–1006. International Foundation for Autonomous Agents and Multiagent Systems, 2008.

[6] J. F. Banzhaf. Weighted voting doesn't work: a mathematical analysis. *Rutgers Law Review*, 19:317–343, 1965.

[7] A.P. Dawid and A.M. Skene. Maximum likelihood estimation of observer error-rates using the em algorithm. *Applied Statistics*, pages 20–28, 1979.

[8] O. Dekel, F. Fischer, and A.D. Procaccia. Incentive compatible regression learning. In *SODA*, 2008.

[9] P. Everaere, S. Konieczny, and P. Marquis. The strategy-proofness landscape of merging. *Journal of Artificial Intelligence Research*, 2007.

[10] L.S. Gottfredson. Why g matters: The complexity of everyday life. *Intelligence*, 24(1):79–132, 1997.

[11] A.R. Jensen. The g factor: The science of mental ability. *London: Westport*, 1998.

[12] G. Kasneci, J. Van Gael, R. Herbrich, and T. Graepel. Bayesian knowledge corroboration with logical rules and user feedback. In *Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases: Part II*, pages 1–18. Springer, 2010.

[13] D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[14] Dennis L. Voting power in the governance of the international monetary fund. *Annals of Operations Research*, 109(1-4):375–397, 2002.

[15] C. List and R.E. Goodin. Epistemic democracy: generalizing the condorcet jury theorem. *Journal of Political Philosophy*, 9(3):277–306, 2001.

[16] C. List and C. Puppe. Judgment aggregation: A survey. *Handbook of Rational and Social Choice*, 2009.

[17] J. Lorenz, H. Rauhut, F. Schweitzer, and D. Helbing. How social influence can undermine the wisdom of crowd effect. *Proceedings of the National Academy of Sciences*, 2011.

[18] D. Lubinski. Introduction to the special section on cognitive abilities: 100 years after spearman's (1904)"general intelligence,'objectively determined and measured". *Journal of Personality and Social Psychology*, 86(1):96, 2004.

[19] J.A. Lyle. Collective problem solving: Are the many smarter than the few? 2008.

[20] A. McLennan. Consequences of the condorcet jury theorem for beneficial information aggregation by rational agents. *American Political Science Review*, pages 413–418, 1998.

[21] J. Pearl. *Probabilistic reasoning in intelligent systems : networks of plausible inference*. 1988.

[22] D.M. Pennock and R. Sami. Computational aspects of prediction markets, 2007.

[23] J.C. Raven. Standard progressive matrices plus, sets a-e.

[24] J.C. Raven. *Progressive matrices*. Éditions Scientifiques et Psychotechniques, 1938.

[25] J.C. Raven. The raven's progressive matrices: Change and stability over culture and time. *Cognitive Psychology*, 41(1):1–48, 2000.

[26] J.C. Raven, J.H. Court, and J.E. Raven. *Manual for Raven's progressive matrices and vocabulary scales*. HK Lewis, 1978.

[27] D. Samet, Y. Tauman, and I. Zang. An application of the aumann-shapley prices for cost allocation in transportation problems. *Mathematics of Operations Research*, pages 25–42, 1984.

[28] F.L. Schmidt and J. Hunter. General mental ability in the world of work: occupational attainment and job performance. *Journal of Personality and Social Psychology*, 86(1):162, 2004.

[29] A. Sen. Social choice theory. *Handbook of mathematical economics*, 3:1073–1181, 1986.

[30] L. S. Shapley. A value for n-person games. *Contrib. to the Theory of Games*, pages 31–40, 1953.

[31] L. S. Shapley and M. Shubik. A method for evaluating the distribution of power in a committee system. *American Political Science Review*, 48:787–792, 1954.

[32] C. Spearman. The abilities of man. 1927.

[33] S.H. Tijs and T.S.H. Driessen. Game theory and cost allocation problems. *Management Science*, pages 1015–1028, 1986.

[34] P. Welinder, S. Branson, S. Belongie, and P. Perona. The multidimensional wisdom of crowds. In *Neural Information Processing Systems Conference (NIPS)*, volume 6, page 8, 2010.

[35] A.W. Woolley, C.F. Chabris, A. Pentland, N. Hashmi, and T.W. Malone. Evidence for a collective intelligence factor in the performance of human groups. *Science*, 330(6004):686, 2010.

Efficient Opinion Sharing in Large Decentralised Teams

Oleksandr Pryymak, Alex Rogers, Nicholas R. Jennings
Electronics and Computer Science
University of Southampton
Southampton, UK
{op08r, acr, nrj}@ecs.soton.ac.uk

ABSTRACT

In this paper we present an approach for improving the accuracy of shared opinions in a large decentralised team. Specifically, our solution optimises the opinion sharing process in order to help the majority of agents to form the correct opinion about a state of a common subject of interest, given only few agents with noisy sensors in the large team. We build on existing research that has examined models of this opinion sharing problem and shown the existence of optimal parameters where incorrect opinions are filtered out during the sharing process. In order to exploit this collective behaviour in complex networks, we present a new decentralised algorithm that allows each agent to gradually regulate the importance of its neighbours' opinions (their social influence). This leads the system to the optimised state in which agents are most likely to filter incorrect opinions, and form a correct opinion regarding the subject of interest. Crucially, our algorithm is the first that does not introduce additional communication over the opinion sharing itself. Using it 80-90% of the agents form the correct opinion, in contrast to 60-75% with the existing message-passing algorithm DACOR proposed for this setting. Moreover, our solution is adaptive to the network topology and scales to thousands of agents. Finally, the use of our algorithm allows agents to significantly improve their accuracy even when deployed by only half of the team.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Reliability

Keywords

Self-organisation, Emergent behaviour, Distributed problem solving

1. INTRODUCTION

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

The problem of sharing information in large networked teams of hundreds and thousands of agents has recently received much attention in terms of how to facilitate the resolution of conflicting information and to improve its accuracy. In this paper we focus on a case when agents share *opinions* about the state of the common subject of interest, and these opinions may conflict. The aim of each agent is to maximise its *accuracy* by forming only *correct* opinions that correspond to the subject's true state. To fulfil its aim, the agent fuses opinions from other agents and forms its own opinion. However, in many decentralised systems, such as social communities and sensor networks, agents' interactions are restricted by a communication network. Thus, agents can receive opinions only from a limited number of their network neighbours. The complex topological properties of such communication networks [8] give rise to surprising and non-trivial *collective behaviour* in these opinion sharing processes [3, 10]. For example, a team may suddenly change its state when a large number of agents change their opinions in an *opinion cascade* after just a single new observation has been introduced [1]. Therefore, there is a crucial need to take into consideration and exploit the properties of collective behaviour in developing an agent-based approach for improving the accuracy of shared opinions.

Recently, Glinton, Scerri and Sycara [4, 5, 6] have presented an agent-based model of opinion sharing to analyse the impact of collective behaviour on the accuracy of the agents' opinions. In contrast to the classical models of opinion sharing [3], the researchers model observations of the common subject of interest by agents with noisy sensors. This approach enables us to reason about the accuracy of the opinions. Their analysis reveals that the accuracy dramatically increase in a narrow range of *social influence parameters* that encode how agents affect each other [4]. This narrow range correspond to a phase transition between a stable state of the team (where opinions are not shared) and an unstable one (where early and possible incorrect opinions are shared on a large scale). Close to this phase transition the number of agents that take part in an opinion cascade is distributed by a power law, and thus, opinion sharing exhibits *scale-invariant dynamics*. At this point, frequent small cascades prevent the team from overreacting to early and possibly incorrect opinions. While less frequent, large cascades share the locally supported opinions to the rest of the team. In Glinton et al.'s model the social influence parameters, on which the opinion sharing depends, are implemented as *importance levels* that each agent attributes to its neighbours' opinions. Unfortunately, it is impossible to predict the im-

portance levels that introduce these scale-invariant dynamics since the properties of the communication network has a significant influence on the sharing processes and thus, analytical analysis cannot be applied to teams with complex communication networks [2]. In order to achieve the optimal parameter in such system, Glinton et al. proposed the Distributed Adaptive Communication for Overall Reliability (DACOR) algorithm [5]. DACOR is an online algorithm that adjusts the agents' importance levels according to the estimated local *branching factor* – the expected number of neighbours that would change their opinions following the change of an agent's opinion. In particular, it was found that in the area of optimal parameters the branching factor is close to 1.

However, actually performing a decentralised estimation of the branching factor by DACOR requires significant communication overhead compared to the opinion sharing itself. In many settings the capabilities of the agents are restricted and communication is limited to opinion sharing only. These restrictions can be found in many realistic settings, such as sensor networks where it is expensive to share data, or social communities where people rely on the opinions of others when they do not have enough resources or skills to analyse the original information themselves. Therefore, there is a need to address the open problem of improving the accuracy of shared opinions in settings where communication is strictly limited to opinion sharing. Moreover, to be applicable across a broad range of domains, a solution must adapt to the network topology. However, as our empirical evaluation reveals, the internal parameters of DACOR are very sensitive to the team's configuration and they have to be tuned individually for different domains.

To address these shortcomings, we present a decentralised algorithm for Adaptive Autonomous Tuning (AAT) of agents' importance levels. AAT improves the accuracy of the opinions in complex networks without introducing additional communication overhead. In contrast to DACOR, our algorithm relies solely on agents' local observations, rather than resource-intensive estimation of the branching factor. Our approach is based on the observation that opinions in the team becomes dramatically more accurate when the agents apply the minimal importance levels to their neighbours that still enable them to share opinions on the team scale. By meeting this condition at the individual agent level, AAT gradually tunes the team to the phase transition in the dynamics of the opinion sharing between the stable and the unstable state. In such settings the team exhibits significant improvement in the accuracy of agents' opinions since the team does not overreact to early and possibly incorrect opinions and the agents share opinions in smaller groups before a large cascade occurs.

In more detail, the contributions of this paper are:

1. We develop a novel decentralised algorithm, AAT, that improves the accuracy of the opinions in a large team with a complex communication network by exploiting the properties of its collective behaviour. Crucially, AAT is the first solution that operates when communication is strictly limited to opinion sharing, and is able to adapt to the specific communication network in which the agents find themselves.
2. We empirically evaluate AAT and show that it significantly outperforms the state-of-the-art solution, DACOR. Specifically, using AAT, 80-90% of the agents' typically form the correct opinion about the common subject of interest. This figure is significantly higher than 60-75% for DACOR, and close to 90-95% that can be reached by pre-tuning a team by an expensive empirical exploration of its parameters. Moreover, AAT introduces less computation expenses and each agent requires 10^4 times less actions than with DACOR.
3. We show that AAT is the first efficient solution designed to improve accuracy in teams with indifferent agents that do not participate in the optimisation process. Specifically, it significantly improves the accuracy with up to 50% of indifferent agents in the team. This implies that AAT potentially can be used by the large teams where it is impossible to update the behaviour of all agents, such as human-agent networks or heterogeneous sensor networks.

The remainder of this paper is organised as follows. In Section 2 the model of the environment, its properties and metrics are discussed. In Section 3 the agents' dynamics are analysed and AAT is presented. Then, in Section 4 AAT is empirically evaluated to demonstrate its efficiency in contrast to DACOR and it is compared with a team pretuned for the highest accuracy. Section 5 concludes this work.

2. PROBLEM DESCRIPTION

In this section, we formally describe an agent-based model of opinion sharing that was recently proposed and analysed by Glinton, Scerri and Sycara [4, 5, 6]. The aim of the model is to capture the complex dynamics of opinion sharing in a network of cooperative agents. In this model, some agents have access to noisy sensors, and they introduce to the team conflicting opinions of which only one is correct. Due to communication constraints agents can only share opinions with their network neighbours, without any additional information.

2.1 Model of Opinion Sharing

Formally, the Glinton, Scerri and Sycara model consists of a large set of agents $A = \{i^l : l \in 1 \dots N\}$, $N \gg 100$ connected by a undirected network $G(A, E)$ where E is the set of edges indicating which agents are neighbours and can therefore communicate. Each agent, $i \in A$ has a neighbourhood $D_i = \{j : \exists\,(i,j) \in E\}$ and the average number of neighbours is defined as the expected degree d, where $d = \sum_{i \in A} |D_i|/N$. We assume that the network is sparse $d \ll N$ in order to observe the cascading behaviour in the sharing process.

The aim of every agent, and eventually of the whole team, is to find the true state b of the common subject of interest, for example $B = \{\texttt{white}, \texttt{black}\}$, where $b \in B$. We support the assumption that B is binary following the argument that a binary choice can be applied to a wide range of real world situations [11]. However, our approach, presented later, does not rely on this limitation and can be extended for $|B| > 2$.

The goal of each agent is to form its own correct *opinion*, o_i, such that $o_i = b$. To recover this true state, agents rely on noisy sensors and their neighbours' opinions about the value of b. To decide which conflicting opinion to adopt, agent i forms its private belief $P_i(b{=}\texttt{white})$, which is the probability that $b = \texttt{white}$ (further denoted as P_i) and consequently $1 - P_i$ is the probability of $b = \texttt{black}$. The agent updates its belief starting from some initial prior P_i' and the ongoing

belief is denoted by P_i^k where k is the current step of the belief update sequence.

Only a small subset of agents $S \subset A$, $|S| \ll N$ have noisy sensors and can make observations of the true state b. Each agent with a sensor $i \in S$ periodically receives an observation $s_i \in B$ with a low accuracy r $(0.5 < r \ll 1)$, which is the probability of returning the true state b. To incorporate a new observation from the sensor into its belief, the agent uses formal reasoning based on Bayes' theorem:

$$P_i^k = \frac{c_{\text{upd}} P_i^{k-1}}{(1-c_{\text{upd}})(1-P_i^{k-1}) + c_{\text{upd}} P_i^{k-1}}, \quad (1)$$

$$\text{where} \begin{cases} c_{\text{upd}} = r & \text{if } s_i = \texttt{white} \\ c_{\text{upd}} = 1-r & \text{if } s_i = \texttt{black} \end{cases}$$

After updating its belief with a number of observations the agent may become confident enough to form its own opinion o_i^k about the true state b. It does so once its belief P_i^k exceeds thresholds, following the opinion update rule:

$$o_i^k = \begin{cases} \texttt{undeter.}, & \text{initial, if } k{=}0 \\ \texttt{white}, & \text{if } P_i^k \geq \sigma \\ \texttt{black}, & \text{if } P_i^k \leq 1{-}\sigma \\ o_i^{k-1} & \text{otherwise} \end{cases} \quad (2)$$

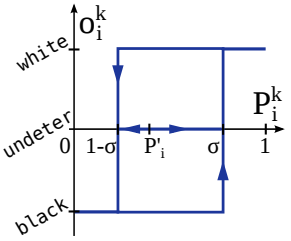


Figure 1: The opinion update rule

where thresholds $\{1{-}\sigma, \sigma\}$ are the *confidence bounds* and $0.5 < \sigma < 1$. The opinion update function has the shape of a sharp hysteresis loop (Fig. 1), and because sensors are noisy, it is possible that later observations will support the opposite opinion, and the agent may change its opinion.

Every time the agent changes its opinion, it communicates the new opinion to its neighbours. Consequently, these neighbours update their own beliefs and may form their own opinions. If the agent changes its opinion following a received opinion from its neighbour, it participates in an *opinion cascade* where a number of agents change their opinions in a sequence after a critical sensor observation. In order to incorporate opinions of the neighbours, the agent uses Bayes' theorem to update its belief similarity to sensor observations, such that when the agent receives new opinions from its neighbours $\{o_j : j \in D_i\}$, it uses the same belief update rule for each received opinion o_j:

$$\text{Eq. (1), where} \begin{cases} c_{\text{upd}} = t_i & \text{if } o_j = \texttt{white} \\ c_{\text{upd}} = 1-t_i & \text{if } o_j = \texttt{black} \end{cases} \quad (3)$$

where $t_i \in [0,1]$ is the *importance level*. This is the measure of the social influence of the neighbour's opinion (that is a conditional probability on opinions communicated from the neighbours). Note, the similarity with Equation 1 such that the importance level is analogous to the accuracy of a noisy sensor, r. However, unlike the accuracy r of a sensor, importance level t_i is unknown and each agent must find its value. In Section 3 we offer our algorithm for this purpose.

The agents in this model are cooperative and thus, they consider only the range $t_i \in [0.5, 1]$, where $t_i = 0.5$ indicates that the received opinion is ignored, and $t_i = 1$ is the maximum importance such that the agent changes its belief to $P_i^k = \{1, 0\}$ (depending on the received opinion) regardless of its previous value P_i^{k-1}. The model implies that the neighbours can be equally wrong in their opinions since sensor readings are introduced randomly. Therefore, it makes an additional assumption that the agent does not differentiate the sources of received opinions and applies the same importance level t_i for all its neighbours. We intend to relax this assumption in our future work and develop techniques that will help to make decisions about the importance of the opinions of each neighbours' individually.

Glinton et al. showed that this model exhibits emergent behaviour and the agents' opinions converge to the true state dramatically more often when the number of agents that take part in an opinion cascade is distributed by a power law, that is known as scale-invariant dynamics [5]. The importance levels are a key parameter which regulate the sharing process and thus, impact the distribution of sizes of opinion cascades. Unfortunately, it was shown that it is infeasible in the general case to predict the importance levels (t_{emr}), at which the emergent behaviour occurs, as this is highly dependent on topology of the network, the distribution of the priors of the agents' and the properties of the sensors. When the team operates with importance levels lower than the critical $\forall i \in A : t_i \ll t_{\text{emr}}$, it is in the stable state of its dynamics and the agents cannot form their own opinions because their beliefs never cross the confidence bounds. Conversely, the team is in the unstable state when $t_i \gg t_{\text{emr}}$, and the agents instantly form confident beliefs, propagate the first, possibly incorrect opinion, and do not benefit from the presence of multiple sensors in the team.

2.2 Performance Metrics of the Model

In order to measure the performance of the team, the model is simulated for a number of opinion dissemination rounds, $M = \{m_l : l \in 1 \ldots |M|\}$, where in each round the new true state $b^m \in B$ is selected randomly. We observe the agents' final opinions, o_i^m, at the end of each round, m. Each round is limited by a large number of belief update steps, k, after which the team is likely to converge to the state where no agent is willing to change its opinion. The end of each round constitutes a certain deadline when the current true state expires. It may be followed by further rounds, in which case, the agents reset their beliefs and opinions to the initial values.

To measure the average accuracy of the agents' opinions at the end of each round, Glinton et al. [5] proposed a metric based on the accuracy of the team that was defined as the ratio between the number of dissemination rounds when the agents' final opinions are correct versus incorrect. This metric heavily penalises the team for disseminating incorrect opinions. However, it can be also maximised if a large proportion of the team does not form any opinion. This is somewhat problematic because we note that in many scenarios it is also important for the agents to form an opinion even if that opinion turn out to be incorrect. Thus, there is a need to balance both the need to be correct, and to actually form an opinion. Therefore we offer the accuracy

metric that measures how often an agent forms the correct opinion on average:

$$R = \frac{1}{N|M|} \sum_{i \in A} |\{m \in M : o_i^m = b^m\}| \cdot 100\% \quad (4)$$

Additionally we introduce a metric from a perspective of a single agent. Since it cannot determine when it has formed a correct opinion, the agent is interested to measure how often it forms an opinion. We denote this as an agent's *awareness rate*, h_i, that is the proportion of dissemination rounds where the agent i held an opinion rather than being undetermined compared to the total number of rounds:

$$h_i = \frac{|\{m \in M : o_i^m \neq \texttt{undeter.}\}|}{|M|} \quad (5)$$

This myopic metric can be calculated locally by each agent and we use it as a basis of our algorithm later. Having introduced the model, we look next at algorithms which optimise the accuracy R, and in Section 4 we offer additional metrics to evaluate their efficiency.

3. AUTONOMOUS ADAPTIVE TUNING

In this section, we present our Autonomous Adaptive Tuning (AAT) algorithm, for improving the accuracy R of a complex communication network by exploiting its collective behaviour. In contrast to the existing algorithm, DACOR, our solution does not introduce communication overhead and communication is strictly limited to opinion sharing. Specifically, in order to estimate the local branching factor, DACOR requires that following a change of an agent's opinion, all its neighbours communicate on average d^2 additional service messages, where d is the expected number of neighbours.

We address this shortcoming by developing a new solution that updates agents' importance levels autonomously, relying on their local observations. Specifically, AAT is built on the observation that accuracy significantly increases when the dynamics of the opinion sharing is in the phase transition between the stable state (when opinions are not shared, $\forall i \in A : h_i \ll 1$) and an unstable one (when the first introduced opinion is propagated on a large scale, $h_i = 1$). This creates a condition where the team does not overreact to incorrect opinions and the agents share opinions in smaller groups before a large cascade occurs. To reach this area of optimal parameters, AAT gradually tunes an importance level of each agent individually.

The three stages of AAT are described in the following sections. Firstly each agent running AAT builds a set of candidate importance levels to reduce the search space for the following stages. Then the agent estimates the awareness rates of the candidate levels after each dissemination round. Finally, the agent selects an importance level to use in the following round, considering how close its estimated awareness is to the target awareness rate.

3.1 Candidate Importance Levels

In this section, we discuss how each agent running AAT selects a number of candidate importance levels, T_i, which reduces the continuous problem of selecting an importance level to use, t_i, from the range $[0.5, 1]$ to a discrete problem. In the general case T_i may be populated with importance levels drawn from the range $[0.5, 1]$ with a given step size, for example 0.01. However, by analysing the dynamics of an agent we can offer a solution for selecting a smaller number of candidate levels which will help AAT to converge to the optimal parameters faster.

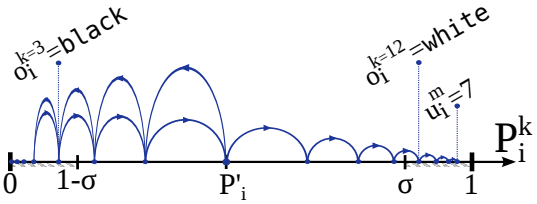


Figure 2: The sample dynamics of an agent's belief with marked steps when the agent changed its opinion.

Since the number of sensors is very small, we focus on the analysis of the agents without sensors who inform their beliefs using only their neighbours' opinions. For example, Figure 2 illustrates the sample dynamics of an agent's belief, P_i^k, where the agent i participated in 2 opinion cascades of conflicting opinions. Starting from its prior P_i', using Bayes' theorem the agent updates its belief with 4 neighbours' opinions that support '`black`' (i.e. 4 updates to the left from the prior P_i' that decrease the agent's belief $P_i^k(b = \texttt{white})$), after which the agent sequentially receives 11 opinions supporting '`white`' (i.e. updates to the right that increase P_i^k).

Clearly, the most important moments in this dynamic are the update steps when the agent change its opinion (steps $k = 3$ and $k = 12$) since only at these steps does the agent communicates a new opinion to its neighbours. To find all the cases whereby the agent can influence the local dynamics, we must find all the importance levels for which the opinion formation process may change. According to the opinion update rule (Eq. 2) we can limit this analysis only to those cases when the agent's belief coincides with one of the confidence bounds $P_i^k \in \{\sigma, 1\text{–}\sigma\}$. Considering also that the maximum number of opinions that the agent can receive is limited to the number of its neighbours, $|D_i|$, we can allow each agent to pre-calculate the candidate importance levels. Specifically, the agent has to find only those importance levels for which its belief coincides with one of the confidence bound $P_i^l \in \{\sigma, 1\text{–}\sigma\}$ in $l \in 1 \ldots |D_i|$ updates (see Eq. 3). By solving this problem, the agent constructs a set of the candidate importance levels that lead to opinion formation after receiving $1 \ldots |D_i|$ identical opinions and reaching the confidence bound σ or $1 - \sigma$:

$$T_i = \left\{ t_i^l : P_i^l(t_i^l) = \sigma,\ l \in 1...|D_i| \right\} \cup \left\{ t_i^l : P_i^l(1 - t_i^l) = 1 - \sigma,\ l \in 1...|D_i| \right\} \quad (6)$$

As a result, the set of candidate levels is limited to twice the number of neighbours: $|T_i| = 2|D_i|$. This is a complete set of importance levels for which the agent forms an opinion on different update steps and it has to be initialised only once. Now, the agent has to form its preferences over these candidate levels to select the most appropriate one to use.

3.2 Estimation of the Awareness Rates

In this section we present the criteria according to which AAT selects an importance level from the candidates. As mentioned earlier, AAT is based on our observation that the accuracy, R, is maximised when the dynamics of opinion sharing is in a phase transition between stable and unstable state. In order to reach such optimal parameters the agents

should use the minimal importance levels to their neighbours that still enable them to share opinions on the team scale.

The intuition is that in order to form an accurate opinion, the agent has to gather as many of its neighbours' opinions as possible before forming its own opinion. To do so, it has to use the minimal importance level from its candidate set. However, if all agents use the minimal importance level and wait until all their neighbours form opinions, a deadlock results where the opinion sharing stops. Therefore, each agent must apply a minimal importance level to the received opinions which guarantees that the agent actually forms its own opinion and shares it further.

In terms of the model we can formalise this, such that in order to maximise the accuracy, R each agent has to:

- Form its opinion, and thus, reach a high level of its awareness rate (h_i, the proportion of the rounds where the agent held an opinion rather being undetermined) since the agents with undetermined opinions decrease the team's accuracy;
- Form the correct opinion given its local view. Following the intuition above, in order to do so, the agent has to form an opinion as late as it is possible to gather the maximum number of neighbours' opinions.

To meet these conditions, the agent has to use the minimal importance level out of the candidates, $t_i^l \in T_i$, that always lead to an opinion formation ($h_i = 1$).

However, since sensors introduce observations randomly, the opinion sharing dynamic in the area of the phase transition exhibits stochastic behaviour. As a result, during some rounds opinions are not shared on a large scale and the agents' awareness rates suffer. Therefore, to improve the overall accuracy and find the exact position of the phase transition, each agent i has to compromise its own awareness rate, h_i. Specifically, the agent has to find the minimal importance level, t_i^l out of candidates T_i that delivers the *target awareness rate*, h_{trg}, that is slightly lower than the maximum, 1. Formally, each agent solves the following optimisation problem:

$$t_i = \arg\min_{t_i^l \in T_i} |h_i(t_i^l) - h_{\text{trg}}| \tag{7}$$

where $h_i(t_i^l)$ is the awareness rate that the agent achieves using importance level t_i^l. We analyse the impact of the specific value of h_{trg} on the accuracy in the empirical evaluation (Section 4.1).

Now, in order to perform the optimisation in Equation 7, the agent needs to calculate all awareness rates, $h(t_i^l)$, that would be achieved by using $t_i^l \in T_i$. However, according to the definition of the awareness rate, h_i (Eq. 5), it can be measured only for the importance level, t_i, that the agent currently uses. By analysing the process of the agents' belief update, we propose the following approach to estimate the awareness rate based on the local observation. Specifically, to estimate the awareness rate, $\hat{h}_i^l \approx h(t_i^l)$, the agent has to decide if its opinion could have been formed had it used an importance level, t_i^l, rather than the actually used t_i. We identify two cases that indicate this:

1. Consider the case that the agent used importance level t_i in round m and an opinion was formed, $o_i^m \neq$ undeter. According to the belief update function (Eq. 3) all higher importance levels, $t_i^l \geq t_i$, would have led to the more confident belief ($|P_i(t_i^l)| > |P_i(t_i)|$), and thus, to opinion formation.

Algorithm 1 AAT

Procedure UPDATE(i)
{Revises the current importance level after each round}
1: **if** OPINIONS RECIEVED : $u_i^m \neq 0$ **then**
2: **for all** CANDIDATE LEVELS : $t_i^l \in T_i$ **do**
3: **if** OPINIONFORMED(t_i^l, t_i, m) = `True` **then**
4: $\hat{h}_i^l$ = UPDATEAVERAGEAWARENESS$(\hat{h}_i^l,\ 1)$
5: **else**
6: $\hat{h}_i^l$ = UPDATEAVERAGEAWARENESS$(\hat{h}_i^l,\ 0)$
7: t_i = SELECTBYAWARENESS $\left(\langle t_i^l,\ \hat{h}_i^l\rangle : l \in 1..|T_i|\right)$

2. Otherwise, if the opinion was not formed, the agent can make a decision by comparing the number of updates it has observed and the number required for the candidate level, t_i^l, to form an opinion. Specifically, the minimal number of belief updates required to form the opinion with the candidate level, t_i^l, can be calculated by recursively updating the agent's belief (see Eq. 3) starting its prior until it exceeds one of the confidence bounds: σ for updates with t_i^l, or $1-\sigma$ with $1-t_i^l$. We denote this function as $u(t_i^l, P_i', \sigma)$. At the same time, during the dissemination round the agent can observe the maximum number of updates it has made in favour of any conflicting opinion starting its prior. We denote this value as u_i^m. In Figure 2 it is observed on the last belief update step $u_i^m = |4-11| = 7$. Finally, the opinion should have been formed when the the number of updates required for the candidate t_i^l is smaller or equal than the observed number u_i^m.

Combining these cases, we construct a boolean function that returns `True` if the agent might have formed an opinion in the current round, m using importance level t_i^l with actual importance level t_i:

$$\text{OPINIONFORMED}(t_i^l, t_i, m) = \left(o_i^m \neq \texttt{undeter.} \ \wedge\ t_i^l \geq t_i\right) \vee\ u_i^m \geq u(t_i^l, P_i', \sigma) \tag{8}$$

Following the definition of the awareness rate (Eq. 5), to estimate the awareness rates for the candidate levels the agent has to measure the proportion of dissemination rounds $m \in M$ for which the condition above was matched:

$$\hat{h}_i^l = \frac{|\{m \in M : \text{OPINIONFORMED}(t_i^l, t_i, m) = \texttt{True}\}|}{|M|} \tag{9}$$

Algorithm 1 describes the core procedure of AAT that implements this approach to estimate awareness rates and is executed after each dissemination round. If no opinions were received ($u_i^m = 0$), the agent cannot form its own opinion with any of the importance level, and thus this case is limited by the condition on line 1. In lines 2-6, AAT updates the estimates of the awareness rate for each of the candidate levels according to the procedure described above. Now, according to optimisation problem the agent solves (Eq. 7), it has to select the importance level (line 7) that delivers the awareness rate closest to the target, h_{trg}, considering the high interdependence between agents' choices.

3.3 Strategy to Select an Importance Level

The agents' opinions are highly interdependent and an importance level chosen by a single agent eventually affects the dynamics and awareness rates of all agents. Therefore, if the

agent would greedily select its importance level according to the definition of its optimisation problem (Eq. 7), it may dramatically change local dynamics. Instead, the agent has to employ a strategy with less dramatic changes in its dynamics, in order that entire team estimate awareness rates more accurately and converge to the solution faster.

To construct such a strategy, we note that since the lowest importance level, t_i^1, from the candidates in ascending order, requires more sequential updates to cross one of the confidence bounds, while the largest $t_i^{\max}$ requires less, then the awareness rates are distributed as a hill with a peak for the largest importance level, $t_i^{\max}$. Therefore, we offer a *hill-climbing strategy* that makes use of this observation. If the awareness rate delivered by the currently used importance level, $t_i = t_i^l$, is lower than target $\hat{h}_i^l < h_{\text{trg}}$, the agent must increase the importance level to the closest larger one (i.e. $l = l + 1$). Conversely, if the closest lower importance level is estimated to deliver an awareness rate higher than $\hat{h}_i^{l-1} > h_{\text{trg}}$, the agent chooses to use it in the next round (i.e. $l = l - 1$). Our empirical evaluation confirmed that the hill-climbing strategy delivers the higher accuracy then the greedy strategy and for brevity we present results only of AAT based on it.

4. EMPIRICAL EVALUATION

To empirically evaluate the performance of AAT and the existing DACOR, we consider a wide range of parameters in order to examine their adaptivity and scalability. Specifically, we evaluate the accuracy of teams with $N \in \{150 \dots 2000\}$ agents on networks with a variable expected degree, $d \in \{4 \dots 12\}$. The maximum size of a team is limited due to the high computational expenses required to empirically pretune a team for the highest accuracy that we use later as a benchmark. We consider the following network topologies widely used in the literature: (a) a connected random network; (b) a scale-free network with clustering factor $p_{\text{cluster}} = 0.7$ [7]; (c) a small-world ring network with $p_{\text{rewire}} = 0.12$ of randomised connections. [9]. New opinions are introduced through a small number of sensors ($|S| = 0.05N$ with accuracy $r = 0.55$) that are randomly distributed across the team. To simulate a gradual introduction of new opinions, only 10% of sensors make new observations after the preceding opinion cascade has stopped. Finally, all agents are initialised with the same confidence bound $\sigma = 0.8$, initial opinion $o_i^0 =$ `undeter`., and individually assigned priors P_i' that are drawn from a normal distribution $\mathcal{N}(\mu = 0.5, s = 0.1)$ within the range of the confidence bounds $(1 - \sigma, \sigma)$.

Before every round m we randomly choose the true state $b^m \in B$. Each round stops after 3000 sensors' observations and sequential opinion cascades. After this number of observations, the opinions of the agents with sensors converge to the true state, and thus, the sharing process stops. The end of each round constitutes a deadline when the current true state expires, and agents reset their beliefs and opinions to the initial values. AAT and DACOR tune the importance levels in the first 150 rounds, then the metrics are measured over the following 150 rounds. Error bars in figures indicate the standard errors across a designated number of network instances in each case.

4.1 Selection of the Target Awareness Rate

We first analyse the performance of our algorithm AAT with

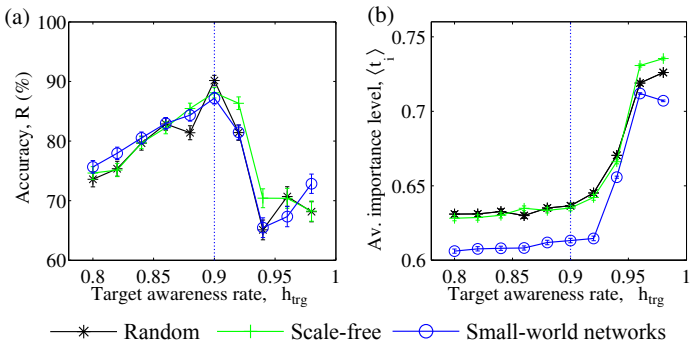


Figure 3: (a) The accuracy and (b) the average importance level achieved by AAT, both depend on the target awareness rate h_{trg} (40 instances of each topology with $N = 1000$ and $d \in \{4 \dots 12\}$).

a regard to its single parameter – the target awareness rate h_{trg}. The analysis supports our earlier assertion, that h_{trg} has to be slightly lower than 1 to tune the team to the area of optimal parameters. Figure 3 shows that the highest accuracy achieved when $h_{\text{trg}} = 0.9$ regardless of the topology of the network. The accuracy significantly drops for the higher values of h_{trg} (Fig. 3a) since agents select much larger importance levels (Fig. 3b) to form opinions out of smaller number of observations. Thus, they become overconfident and the whole team converges to the early opinion without fusing it with later observations that might be more accurate. Considering the results, in our further evaluation we use $h_{\text{trg}} = 0.9$.

4.2 Accuracy of the Opinions

We now benchmark AAT against three alternative solutions. First, we compare against DACOR (with parameters $uA = 10, \gamma = 0.001, \beta = 0.1$ selected to maximise the accuracy of a random network with $d = 8$), the current state of the art solution in this setting. In addition, we also benchmark against a team pre-tuned for the highest performance on a specific network instance. In more detail, to pre-tune a team, we perform a resource intensive empirical exploration of each network instance with fixed importance levels $\forall i \in A : t_i = t$, where $t \in (0.5, 1)$ with a step of 0.05 over $|M| = 150$ rounds. Then we choose the importance level t_{emr} at which the team exhibits the highest accuracy. Note, that this is not the optimal solution, as it is infeasible to explore the whole domain where agents may have different importance levels. Still, this approximation exhibits a high accuracy of 90 to 97% and shows its level that can be achieved by fine tuning. However, t_{emr} varies between different network instances since the area of optimal parameters is very narrow and dependent on the team's configuration. Therefore, to illustrate how difficult such tuning is in practice, we also benchmark against the team with the average $\langle t_{\text{emr}} \rangle$ for the networks of the same size and topology.

The results of the accuracy benchmark are shown in Figure 4a. As can be seen, AAT shows accuracy close to the results of the pre-tuned teams and significantly outperforms the existing solution, DACOR, for all network topologies. AAT scales well, since it reaches the stable accuracy around of 86 to 88% for teams larger than 1000 agents. However, it declines as the team size becomes lower than 1000 agents. This is due to the fact that the properties of collective behaviour are less distinct in smaller teams. Analysis of the

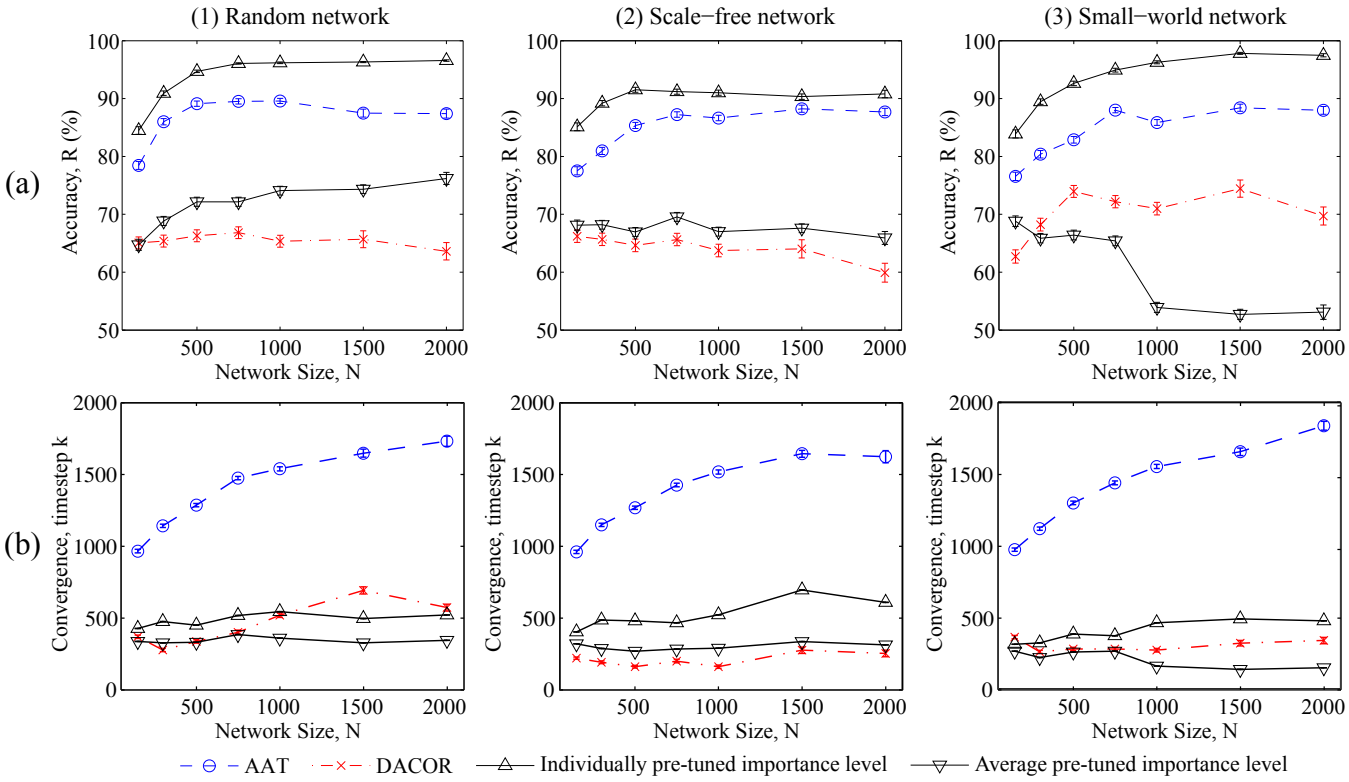


Figure 4: (a) The accuracy and (b) the convergence of a team with AAT, DACOR, and pre-tuned importance levels (40 instances of a each topology and network size with $d \in \{4 \dots 12\}$).

results also show that DACOR, unlike our adaptive AAT approach, is highly dependent on parameters which have to be individually tuned for specific domains. In most cases DACOR has tuned the team to the unstable state where early and possibly incorrect opinions are shared on a large scale. Finally, the low accuracy achieved by teams with $\langle t_{\text{emr}} \rangle$ indicates a clear need for an algorithm such as AAT that can efficiently tune each team individually.

4.3 Opinion Convergence

AAT tunes the team into a phase transition between stable and unstable states where the sharing processes are the slowest that still enable all agents to form their opinions. However, this also implies that agents with AAT may form their opinion slower. To measure the timeliness of the opinion, we offer a *convergence metric* that is the average number of timesteps required for a team to reach the accuracy of $R \geq 80\%$. In order to avoid distortion of its average value, we exclude dissemination rounds when the team did not reach the threshold level of the accuracy.

The results shown on Figure 4b indicate that convergence time for AAT growth steadily with the size of a team. This fact can explained by the increasing sparseness of the network since its degree d is fixed. This results in a slower sharing process as the shortest path increases as well.

DACOR exhibits much faster convergence since it tunes the team into the area of the unstable state. This also explains its low accuracy discussed above. By contrast, most of the teams with the average of pre-tuned levels exhibit stable dynamics and do not share opinions on a large scale, while some are in unstable state that result in a fast convergence.

Finally, the individually pre-tuned teams exhibit relatively fast convergence. This indicates a drawback of AAT that has to be addressed. Specifically, the online approach used to build AAT results in slower convergence and alternative solutions, such as offline pre-tuning, may exhibit equal or higher level of accuracy with significantly faster convergence at the same time.

4.4 Communication and Computation Expenses

AAT is designed to improve accuracy without introducing additional communication over opinion sharing. We compare in Figure 5a the number of messages that agents exchange while the team is tuned by AAT, DACOR, and the minimal number of messages required to share an opinion on a team scale in a single cascade. The latter represents the *minimal communication*, when agents share their opinions only once to the neighbourhood, and thus, communicate in total dN messages. The average number of messages for a team with AAT is similar to the minimal communication, since during some rounds a team does not disseminate opinions on a large scale (as the result of $h_{\text{trg}} < 1$).

In addition, AAT requires radically less actions by the agents that are the changes of the importance levels than DACOR in the process of tuning. AAT updates an importance level only once at the end of each round, while DACOR updates an agent's importance level if any of its neighbours has received new opinion. The results that represent computational expenses, are shown in Figure 5b. Both metrics show that AAT is a highly scalable solution.

4.5 Team with Indifferent Agents

Finally, AAT is robust to the presence of the agents that are *indifferent* and do not participate in the optimisation process. This is due to the fact that the agents running AAT

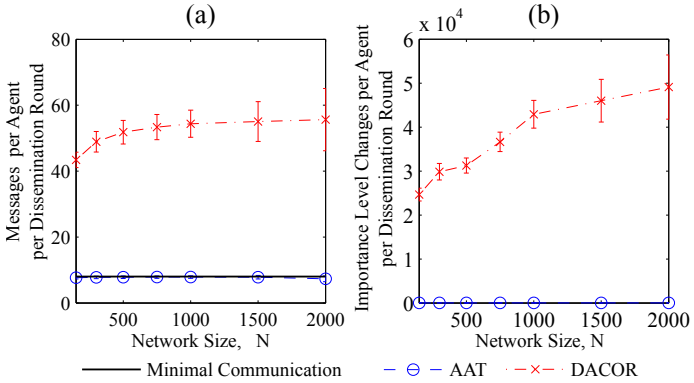


Figure 5: (a) Number of messages and (b) number of importance level changes, for an agent per dissemination round (averaged over all experiments shown in Figure 4).

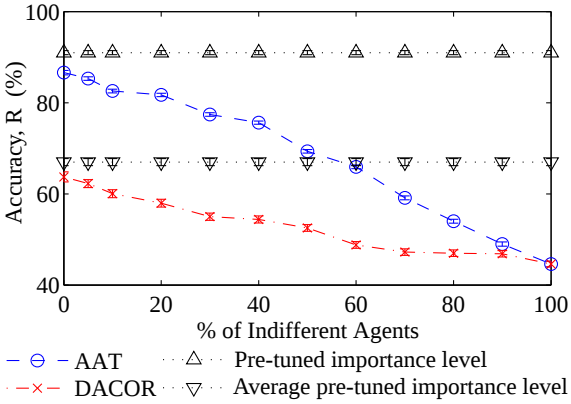


Figure 6: The accuracy of the team with indifferent agents (40 instances of a scale-free network with $N = 1000$, $d \in \{4 \dots 12\}$).

tune their importance levels autonomously by adapting to their neighbourhood. Thus, they mitigate the negative effect introduced by the indifferent agents.

We illustrate this by evaluating a team with a variable number of indifferent agents that are randomly distributed across its population. The importance levels of indifferent agents are not dynamically determined by AAT or DACOR algorithms, but fixed and uniformly selected from the range close to the critical importance level $[0.55, 0.75]$. The results in Figure 6 shows that AAT with up to 50% of indifferent agents delivers higher accuracy than can be achieved by using $\langle t_{\text{emr}} \rangle$. This shows the direct benefit from deploying AAT even on half of the agents in a team over predicting the critical importance level by analysing a number of similar teams. Similar results are obtained for the other topologies and team sizes.

5. CONCLUSIONS

In this paper, we developed a novel decentralised algorithm, AAT, which significantly improves the accuracy of agents' opinions by exploiting the properties of collective behaviour in large networked teams. This is the first solution that can be used by teams with complex communication networks in the settings when communication is strictly limited to opinion sharing. We showed that AAT significantly outperforms the existing algorithm, DACOR, that also introduces additional communication to operate and requires higher computational cost. The accuracy exhibited by AAT is close to the highest accuracy that can be achieved by individually pre-tuning a team by the resource expensive empirical exploration of its parameters. Moreover, we showed that AAT is scalable, adaptive to the team's configuration and robust to the presence of indifferent agents that do not participate in the optimisation process. Finally, since AAT relies only on local view, the importance levels it estimates can be used as the initial trust levels for an elaborate trust models when no additional information is available.

Our future work in this area is to relax an assumption that the agents do not differentiate between their neighbours. This will require a new algorithm that estimates individual importance levels for each neighbour based on their opinion dynamics. Additionally, we also intend to address an outlined problem of developing an attack resistant solution that will help to mitigate the negative influence of malicious agents [6].

6. REFERENCES

[1] S. Bikhchandani, D. Hirshleifer, and I. Welch. A theory of fads, fashion, custom, and cultural change as informational cascades. *Journal of Political Economy*, 100(5):992–1026, 1992.

[2] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. Hwang. Complex networks: Structure and dynamics. *Physics Reports*, 424(4-5):175–308, 2006.

[3] C. Castellano, S. Fortunato, and V. Loreto. Statistical physics of social dynamics. *Reviews of Modern Physics*, 81(2):591–646, May 2009.

[4] R. Glinton, P. Scerri, and K. Sycara. Towards the understanding of information dynamics in large scale networked systems. In *Proceedings of 12th International Conference on Information Fusion (FUSION'09)*, pages 794–801, Seattle, Washington, USA, 2009.

[5] R. Glinton, P. Scerri, and K. Sycara. Exploiting scale invariant dynamics for efficient information propagation in large teams. In *Proceedings of 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pages 21–28, Toronto, Canada, 2010.

[6] R. Glinton, P. Scerri, and K. Sycara. An Investigation of the Vulnerabilities of Scale Invariant Dynamics in Large Teams. In *Proceedings of 10th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'11)*, pages 677–684, Taipei, Taiwan, 2011.

[7] P. Holme and B. Kim. Growing scale-free networks with tunable clustering. *Physical Review E*, 65(2):2–5, Jan. 2002.

[8] J. Leskovec, K. J. Lang, A. Dasgupta, and M. W. Mahoney. Statistical properties of community structure in large social and information networks. *In Proceeding of the 17th International Conference on World Wide Web (WWW'08)*, pages 695–705, 2008.

[9] M. Newman. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4-6):341–346, Dec. 1999.

[10] D. J. Watts. A simple model of global cascades on random networks. *In Proceedings of the National Academy of Sciences of the United States of America*, 99(9):5766–5771, Apr. 2002.

[11] D. J. Watts and P. S. Dodds. Influentials, networks, and public opinion formation. *Journal of Consumer Research*, 34(4):441–458, 2007.

Agents of Influence in Social Networks

Amer G. Ghanem
University of Cincinnati
Cincinnati, Ohio, USA
ghanemar@mail.uc.edu

Srinivasa Vedanarayanan
University of Cincinnati
Cincinnati, Ohio, USA
vedanasn@mail.uc.edu

Ali A. Minai
University of Cincinnati
Cincinnati, Ohio, USA
Ali.Minai@uc.edu

ABSTRACT

In recent years, social networking sites and social media have become a very important part of peoples' lives, driving everything from family relationships to revolutions. In this work, we study the different patterns of interaction behavior seen in an online social network. We investigate the difference in the relative time people allocate to their friends versus that which their friends allocate to them, and propose a measure for this difference in time allocation. The distribution of this measure is used to identify classes of social agents through agglomerative hierarchical clustering. These classes are then characterized in terms of two important structural attributes: Degree distributions and clustering coefficients.

We demonstrate our approach on two large social networks obtained from Facebook. For each network we have the list of all social interactions that took place over six months. The total number of users in the two networks is 939,453 and 841,456, with 1.4 million and 8.4 million interactions, respectively. Our results show that, based the interaction behavior, there are four main classes of agents in both networks, and that they are consistent across the two networks. Furthermore, each class is characterized by a specific profile of degree distributions and clustering coefficients, which are also consistent across both networks.

We speculate that agents corresponding to the four classes play different roles in the social network. To test this, we developed an opinion propagation model where opinions are represented as m-bit strings communicated from agent to agents. An agent receiving an opinion then selectively modifies its own opinions depending on the social and informational value it places upon communications from the sending agent, its overall agreement with the sending agent, and its own propensties. Opinions are injected into the system by agents of specific classes and their spread is tracked by propagating tags. The resulting data is used to analyze the influence of agents from each class in the viral spread of ideas under various conditions. The analysis also shows what behavioral factors at the agent level have the most significant impact on the spreading of ideas.

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

Categories and Subject Descriptors

C.2.4 [**Distributed Systems**]: Distributed Applications

General Terms

Measurement, Performance, Human Factors

Keywords

Social Networks, Agent-Based Models, Information Propagation, Facebook

1. INTRODUCTION

The advent of social networks is a major revolution in the history of the Internet age. Today, being 'online' is the default mode for millions of people worldwide. Many sectors of society such as business, politics, advertising, gaming, etc., have embraced social networks and have sought to exploit their potential to reach the masses. Indeed, social media has allowed even individuals to set up channels of communication that rival the commercial media in their influence. The average user on Facebook has 130 friends, and people spend over 700 billion minutes per month on the site [5]. The growth of users on Google+ has occured even faster. LinkedIn's membership rocketed from 28 million users to more than 35 million users in the six months following the September 2008 onset of economic collapse [6]. The key factor that sustains such social networking sites and the businesses based on them is the propagation of information. Every user in the network is a potential point of idea injection for mass propagation. However, it is clear that not all users can achieve the same level of influence. This depends on many factors such as the users' behavior, the semantic content of the information being propagated, the structural properties of the network, etc. In this paper, we use several large datasets to develop a behavior-based classification of agents in a social network (Facebook), and to evaluate what type of agents are likely to be influential in different situations.

2. BACKGROUND

There is already a large – and growing – body of research devoted to understanding the structure and evolution of online social communities[11][17][1][12]. Some of these studies have classified members of social networks into one of three classes[1][12] based on their structural role in the network: *Singletons* are agents with zero friends; *the giant component* comprises users who are connected directly or indirectly to

a large fraction of the entire network; and the *the middle region* covers small groups whose agents only interact mainly with others within the group and not with the network at large. In this study we aim to classify users into groups based on their *interactional behavior* rather than structural position. We believe that such classification will reveal valuable insights on the way users divide their social time between their friends on the network.

The issue of how ideas spread virally in a social network has been of great interest recently. This work has considered a variety of factors that determine the importance of individual agents in a network. These range from structural properties to the nature of the users and their interactions with each other. Trusov et al. [13] proposed a method for finding influential users in a network community using the number of their logins in a particular time frame as the metric for being "influential". Though intuitively appealing, this approach did not provide any unexpected insights. Similarly, Crandall et al. [3] proposed a method to show that social influence and similarity go hand-in-hand such that users tend to form new links based on similarity, which grows their sphere of social influence and, in turn, creates more similarity with users to whom they create new links.

Simpkins et al. [19] considered the role of psychological and cognitive factors in shaping the profile of idea propagation. They postulated that an idea has a 'cognitive advantage' in being retained or accepted in a particular community which is culturally circumscribed around that idea.

Kempe et al. [8] studied the problem of maximizing information spread in a social community network using recommendation or influential propagation in the form of a decreasing cascade model. In this model, a behavior spreads in a cascading fashion according to a probabilistic rule, beginning with a set of initially "active" or "influential" nodes. In another paper [9], they propose an intuitive greedy algorithm to show that it is a more efficient way to find which set of individuals should be targeted in a network for maximizing the spread of influence. This work is closest to ours in terms of its objectives.

In [10], Kleinberg showed that it is easier to find short chains between points in some networks than others. He proved that networks that include individuals operating with purely local information are very adept at finding these short chains. Considering the dynamics of a network, Ghosh et al. [7] have proposed that predicting influential users depends not only on the structure of the network, but also on details of the dynamic processes occurring on it. They classify processes as conservative and non-conservative, and claim that information spread is non-conservative. They empirically define influence as the number of in-network votes a user's post generates. This influence measure, and the resulting ranking, is evaluated from the dynamics of voting on the social news aggregator Digg, which represents non-conservative information flow. They compare their predictions of different influence models with this empirical estimate of influence. The results show that non-conservative models are better able to predict influential users on Digg and the best predictor metric is found to be the normalized α-centrality.

3. OVERVIEW

This paper presents results from two studies:

Study I: Classification of Social Agents: In this study, we use several anonymized datasets of FaceBook interactions among large groups of agents over extended periods [22] to identify classes of agents based on interaction behavior. In particular, we consider the relative social attention agents devote to interacting with other agents, and identify four distinct types of behavior in this regard. These four classes are found to be robust across multiple datasets in terms of both interaction patterns and structural properties, indicating that they capture real differences betwee agents. We then train a neural network classifier to recognize these classes, and show that it can successfully classify agents in a novel dataset according to their interaction patterns. We also provde provisional interpretations of the four behavior patterns identified in this study.

Facebook Dataset			
Network	Nodes	Edges	Interactions
A:One Month (A1)	431,995	728,243	1,412,252
A:Six Months (A6)	939,453	273,215	7,483,904
A:One Year (A12)	1,164,003	4,555,524	24,373,015
B:One Month (B1)	451,092	827,068	1,974,590
B:Six Months (B6)	841,567	2,513,432	8442,451
B:One Year (B12)	969,047	3,317,531	22,092,564

Table 1: Statisctics of the Facebook Dataset used in this paper

Study II: Simulation of Influence Dynamics: In the second study, we implement an agent-based simulation using a randomly chosen subset of a network in our dataset. Agents in this simulation are labeled according to the class assigned to them in Study I and follow the corresponding pattern of relative social effort in their interactions with other agents. All agents in the system begin with certain "opinions", and communicate these to other agents. This influences the receiving agents to possibly modify their own opinions and to communicate them further. The dynamics of opinions originating in agents of different types are monitored and the relative influence for each type of agent is quantified. Essentially, the question addressed by this study is, "Which class of agents are, on average, the most suitable injection point for an idea that needs to be spread virally?"

4. STUDY I: CLASSIFICATION OF SOCIAL AGENTS

The goal of this study is to classify agents in a large social network into different classes based on their pattern of interaction with other agents. The approach followed is to represent the agents' interaction behavior in a suitable feature space, and to cluster the agents based on these features.

4.1 Datasets:

This study used two datasets - labeled A and B - representing agents on Facebook. Each dataset provides a *social graph* indicating "friend" links between agents, and several *interaction graphs*, indicating contacts between linked agents over a period of time. The statistics for the networks are given in Table 1. The data was obtained from [22] and used with permission.

4.2 Community Extraction

Given the large number of agents in the datasets, we decided to focus, in each network, on a subset of agents that

could be considered to have significant participation in a functionally useful sense. To do this, we extracted *communities* of agents from the social networks using the *clique percolation method (CPM)* [14][4]. In CPM, a *k-community* is defined as the maximal chain of adjacent k-cliques. Two k-cliques are considered to be adjacent if they share *k-1* nodes. We decided to use $k = 5$ because a value of $k > 5$ produces very few communities, and a value of $k = 4$ produces too many small ones. Only agents belonging to at least one community were classified in the analysis below, though their interactions with all agents were taken into account. With this restriction, the number of agents and links analyzed were:

Dataset A1: 1051 nodes and 3859 edges
Dataset A6: 21690 nodes and 202611 edges
Dataset A12: 64879 nodes and 959927 egdes
Dataset B1: 2229 nodes and 8128 edges
Dataset B6: 30532 nodes and 202611 edges
Dataset B12: 53633 nodes and 652776 edges

4.3 The Devotion Measure

The interaction pattern between two connected agents i and j was measured through a quantity termed *relative devotion*, Δ_{ij}, defined as:

$$\Delta_{ij} = (I_{ij}/I_i) - (I_{ji}/I_j) \equiv D_{ij} - D_{ji} \quad (1)$$

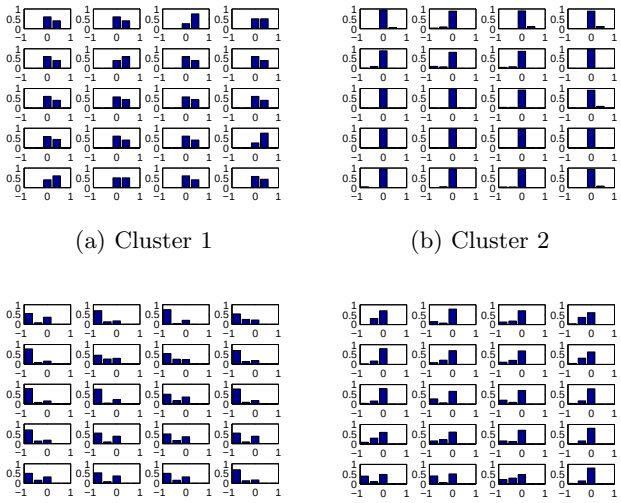
where I_{ij} is the number of interactions that i has with j, I_i is the total number of interactions for i, I_{ji} is the number of interactions that j has with i, I_j is the total number of interactions for j (note that $I_{ij} = I_{ji}$), D_{ij} is the fraction of i's interactions that are with j and D_{ji} the fraction of j's interactions that are with i. Thus, $-1 < \Delta_{ij} < 1$, where a negative value of Δ_{ij} means that i allocates a lower fraction of his/her social effort to interact with j than j is allocating to interact with i, and vice-versa for a positive value. A value of 0 means that i and j allocate the same portion of their social effort to each other.

Based on this definition, each agent i has a *devotion vector*, $\Delta_i = [\Delta_{i1}\ \Delta_{i2}\ ...\ \Delta_{in_i}]$, where n_i denotes the number of agents to which i is directly connected. Since, agents often have high-dimensional devotion vectors with variable lengths across agents, it is convenient to look at the histogram of their relative devotion values. This is obtained by distributing the relative devotion values for i into 5 bins: $[-1.0, -0.6]$, $[-0.6, -0.2]$, $[-0.2, +0.2]$, $[+0.2, +0.6]$ and $[+0.6, +1.0]$. The total is normalized to 1 and the resulting length 5 vector, $q_i = [q_i^1\ q_i^2\ q_i^3\ q_i^4\ q_i^5]$ is called the *feature vector* for agent i. This feature vector constitutes a quantitative representation of the agent in terms of its interaction behavior, and is the basis of classification.

4.4 Agent Clustering

The feature vectors obtained for indivdual agents in Dataset B6 were clustered using an agglomerative hierarchical clustering algorithm with *earth mover's distance*(EMD) as the distance measure. It is defined to be the minimum movement of probability mass required to transform one distribution into another[15]. A fast EMD algorithm developed by Pele et al.[15, 16] was used for calculations. Weighted pair group with averaging was used as the distance measure between merged clusters.

The clustering identified four types of agents in Dataset B6. Some representative feature vectors for each class are shown in Figure 1.



(a) Cluster 1 (b) Cluster 2

(c) Cluster 3 (d) Cluster 4

Figure 1: Feature vectors for 20 representative members of each class.

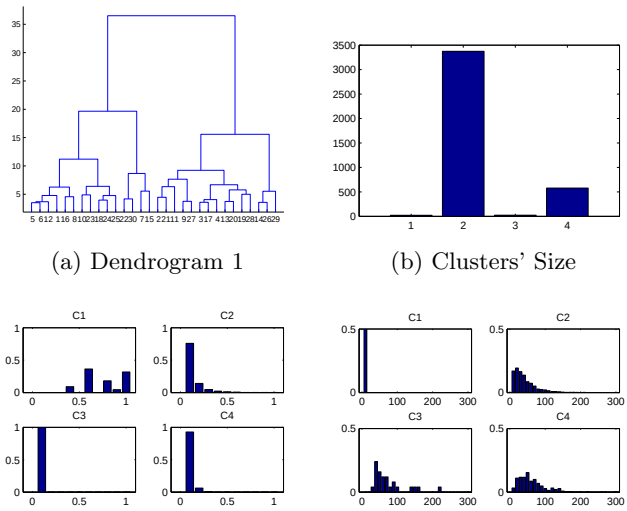
The dendogram for the clustering process is shown in Figure 2(a). Figure 2(b) shows the number of agents assigned to each class, indicating that the vast majority of agents fall into classes 2 and 4. Figures 2(c) and (d) show the distribution of the node clustering coefficient and node degree for each class. Node degree is defined as the number of links incident on the node and the clustering coefficient as the fraction of possible links that exist between the node's immediate neighbors [20][2].

It should be noted that, while only agents belonging to communities were clustered, the calculation of relative devotion, node degree and clustering coefficient used the *full network*, including unclassified agents. Thus, a significant part of the information captured in the distributions of these quantities comes from agents not included in the classification.

4.5 Neural Network Classifier

While clustering provides a useful way to organize the data, it is computationally expensive and, therefore, difficult to use with much larger datasets such as A12 or B12. Even more importantly, it does not provide a way to classify agents other than those included in the clustering process. To address these problems – and to provide further validation of the classes discovered through clustering – a neural network classifier was trained to classify agents based on their feature vectors, using the classes obtained through clustering as the true classes. The classifier had two hidden layers and was trained using the backpropagation algorithm [21, 18] on a portion of the B6 dataset and validated and tested on two other subsets of the data. Figure 3 shows the confusion matrices for the training, validation and testing case as well as over the entire B6 dataset. It is clear the the neural network was extremely successful in learning the classes.

Once trained, the neural classifier was applied to dataset B12 dataset and provided class labels for all 53,633 agents.



(a) Dendrogram 1 (b) Clusters' Size

(c) Clustering Coefficient Distribution (d) Degree Distribution

Figure 2: (a) Dendogram resulting from the clustering process; (b) Number of agents placed in each cluster; (c) Distribution of network clustering coeffiecient for each class; (d) Distribution of node degree for each class.

However, since B12 had not been processed through clustering, the accuracy of these labels could not be verified. For this, we used three methods:

Method 1: In this method, we plotted the feature vectors for 20 representative agents from each class as given by the neural network classifier (Figure 4). A comparison of these with the feature vectors given in Figure 1 shows excellent agreement, indicating that the classes assigned in B12 by the classifier were qualitatively the *same* as those found by clustering in B6. This is strong evidence that these classes are, in fact, robust across different networks, and that the classifier is able to generalize.

Method 2: Next, we plotted the degree distributions (Figure 5) and clustering coefficient distributions (Figure 6) for agents in all four classes as identified in B6 by clustering (Figures 5(a) and 6(a)) and by the neural classifier in B12 (Figures 5(b) ad 6(b)). Comparing these, it is apparent the corresponding distributions are similar, providing further evidence that the clustering and the classifier are finding the same classes in the two networks, and that each of these classes has characteristic distributions of node degree and clustering coefficients.

Method 3: Finally, we also performed a partial direct comparison by taking a subset of agents from B12, subjecting them to the clustering algorithm, and comparing the labels obtained with those given by the neural classifier. The confusion matrix for this comparison is shown in Figure 7, indicating that the two methods agreed on the classification of almost 97% of the agents.

Taken together, these results indicate three things: 1) The interaction behavior of agents in multiple social networks studied falls into four distinct and consistent classes; 2) These classes are robust and have invariant network-based characteristics across different networks; and 3) The neural network classifier is able to assign classes accurately to agents across different networks based on their feature vectors.

(a) Training (b) Validation

(c) Test (d) All

Figure 3: Confusion matrices for the neural network classifier: (a) Data used to train the network; (b) Data used to check for generalization during training but not used for training directly;)c) Data not used during training at all; (d) All data in B6.

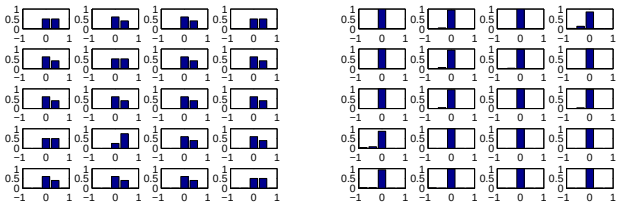
4.6 Interpretation of the Classes

A natural question that arises is whether the classes found by the above analysis are meaningful. While we are still investigating this issue in detail, some provisional suggestions can be made as follows.

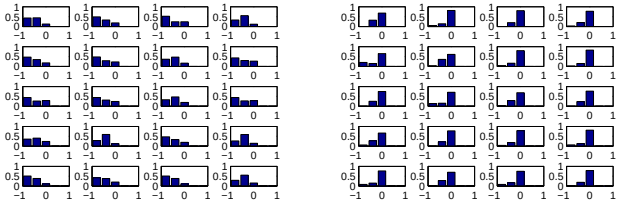
(**Class 1 - Invisibles:** This class comprises a small number of users who have very low node degree, high clustering coefficients, and are only involved in a small number of interactions, suggesting that they belong to a small, tight-knit group of friends. They mainly have positive relative devotion values, which means that their friends are more active than they are, or they are generally ignored by their friends.

Class 2 - Normals: These comprise the majority of the population, and have fairly high degree and a wide range for number of interactions. The majority of Class 2 members have low to moderate clustering coefficients, indicating that they have a broad and loosely knit group of friends, but with significant connectivity among these friends. Their directed devotion values are around 0, which means they interact with friends who interact with them.

Class 3 - Celebs: This class has a small number of members with the majority being high degree nodes. The members of this class have very low clustering coefficients and a wide range of interactions, suggesting that they interact with a large, disparate set of people. Their directed devotion values are strongly negative, indicating that they allocate a smaller portion of their social effort to their friends, who actually allocate a larger portion of their social effort to them. In fact, most of the agents connected to Class 3 agents fall

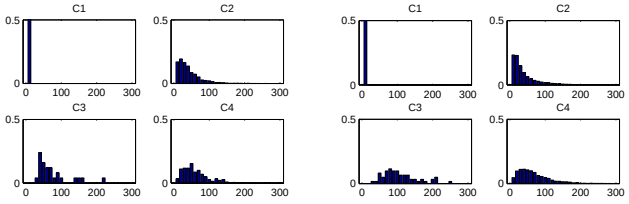


(a) Class 1 (b) Class 2



(c) Class 3 (d) Class 4

Figure 4: Feature distributions of typical members of clases assigned by the neural classifier. This figure should be comparted with Figure 1.



(a) Clustering (b) Classifier

Figure 5: Comparison of degree distribution

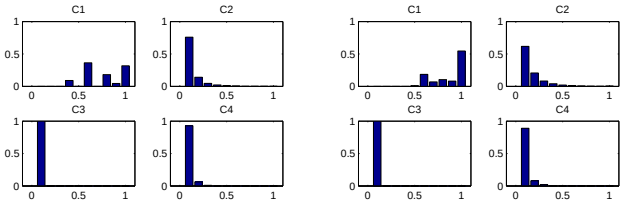
into the unclassified category and have highly positive relative devotion values (not shown).

Class 4 - Casuals: This class has a significant number of members with the majority having degrees in the range of 40-70. Most of them have low clustering coefficients, indicating that they connect with a disparate group of agents. The members of this class have a wide range of intercations and devotion values skewed slightly negative., indicating that they allocate less social effort to their friends than their friends allocate to them.

5. STUDY II: SIMULATION OF INFLUENCE DYNAMICS

A central attribute of social networks is their ability to spread information and influence – a fact used in viral marketing, political campaigning, etc. Given the very interesting agent classification described above, it is natural to ask whether agents of some class(es) are more or less influential. We addressed this issue through simulations of a multi-agent models where the spread of information injected in one agent of a particular class can be tracked across the network over time.

The propagation of influence in a network can be affected



(a) Clustering (b) Classifier

Figure 6: Comparison of clustering coefficient distribution

Cluster ID \ Classification ID	1	2	3	4
1	86	0	0	0
2	0	1940	0	0
3	0	0	61	28
4	0	60	0	472

Figure 7: Direct neural network classifier validation on a subset of B12 data

by a host of factors ranging from the structural characteristics of the network to the nature of the ideas being communicated. Here, we use a very simple model to study how opinions might spread in the type of network analyzed above. We consider the efficacy of agents from particular classes in spreading their opinions within a large network where the interaction behavior of individual agents follow those seen in the corresponding datasets – for both classified and unclassified agents.

5.1 Model Description

The social network used for the simulations is a subset of the A12 network described above, with about $n = 125,000$ agents. These are chosen to include a roughly proportionate sample of classified agents and all their friends, whether classified or unclassified. The numbers for each type are: Class 1 - 108; Class 2 - 40,000; Class 3 - 114; Class 4 - 4,314; Unclassified - 80,464. Each agent's type is assigned as determined by the clustering process (including the type "unclassfied"), and its interaction behavior reflects that seen in the actual dataset. Each agent, i, has a time-varying *opinion vector*, $z_i(t) = [z_{i1}(t) \ ... \ z_{im}(t)]$, with m elements, each indicating an opinion on some issue at time t. Each component z_{ik} can take values -1 (positive); $+1$ (negative), or 0 (don't care). The simulations used a value of $m = 7$, with all bits assigned randomnly and independently with equal probability for the three values.

During simulation, every agent i sends out one message per time-step to a randomly selected *target agent* among its friends. The probability of agent i choosing a target agent, j, at step t is given by $p_{ij}(t) = D_{ij}/\sum_{k\in N(i)} D_{ik}$, where $N(i)$ denotes the set of nodes to which i is directly connected (its circle of friends) and D_{ik} is the fraction of i's interactions that are with k according the real dataset A1, i.e., $D_{ik} = I_{ik}/I_i$. Thus, over time, the fractions of inter-

actions i has with each of its friends reflect the proportion found in the actual dataset. A communication from agent i at time step t comprises its entire current opinion vector, $z_i(t)$. When agent j receives a message from agent i, it identifies all the mismatching components, and independently considers whether to modify its opinion on each of these to match the one received from i. The probability of doing so is determined by three *influencing factors*:

1. The *similarity*, $S_{ij}(t)$ between its own current opinion vector, $z_j(t)$, and the received opinion vector, $z_i(t)$, calculated as the number of matching bits in the two vectors. The probability of change in component k is increased by higher S_{ij}, i.e., agent j is likelier to be influenced by like-minded friends.

2. Its *relative devotion*, Δ_{ji} towards the sending agent, i. The probability of change in component k is increased by higher Δ_{ij}, i.e., agent j is likelier to be influenced by agents it holds in higher esteem (as indicated by relative devotion).

3. The *intrinsic receptiveness*, $R_j^k \in [0,\ 1]$, of agent j for component k, indicating how receptive it is to changing its opinion on this component.

The probability of changing component z_j^k to agree with z_i^k is given by:

$$p_j^k(t) = w_1 f_1(S_{ij}) + w_2 f_2(\Delta_{ij}) + f_3(R_j^k) \tag{2}$$

where the w_l are weights between 0 and 1, with $w_1 + w_2 + w_3 = 1$, and

$$f_1(S_{ij}) = \begin{cases} \frac{1}{2}[1 - (1 - 2S_{ij})^{1-\alpha_j}] & if\ S_{ij} < 0.5 \\ 0.5 & if\ S_{ij} = 0.5 \\ \frac{1}{2}[1 + (2S_{ij} - 1)^{1-\alpha_j}] & if\ S_{ij} > 0 \end{cases} \tag{3}$$

where $\alpha_j \in [0, 1]$ is the *similarity influence parameter*.

$$f_2(\Delta_{ij}) = \begin{cases} \frac{1}{2}[1 - (-\Delta_{ij})^{1-\beta_j}] & if\ \Delta_{ij} < 0 \\ 0.5 & if\ \Delta_{ij} = 0 \\ \frac{1}{2}[1 + \Delta_{ij}^{1-\beta_j}] & if\ \Delta_{ij} > 0 \end{cases} \tag{4}$$

where $\beta_j \in [0, 1]$ is it devotion influence parameter.

$$f_3(R_j^k) = R_j^k = \gamma_j\ \forall k \tag{5}$$

where $\gamma_j \in [0, 1]$ is the *receptiveness influence parameter*.

Thus, the vector $(\alpha,\ \beta_j,\ \gamma_j)$ determine the *personality profile* of agent j in terms of its "influenceability", with each parameter controlling the power of a single factor as follows:

- If $\alpha_j = 0$, $f_1(S_{ij}) = S_{ij}$, i.e., linear dependence on similarity. As α_j increases towards 1, $f_1(S_{ij})$ approaches a threshold function at $S_{ij} = 0.5$, so friends with $S_{ij} > 0.5$ have very high influence on i and friends with $S_{ij} < 0.5$ have almost none.

- If $\beta_j = 0$, $f_2(\Delta_{ij}) = (1+\Delta_{ij})/2$, i.e., linear dependence on devotion. As β_j increases towards 1, $f_2(\Delta_{ij})$ approaches a threshold function at $S\Delta_{ij} = 0$, so friends with $\Delta_{ij} > 0$ have very high influence on i and friends with $\Delta_{ij} < 0$ have almost none.

Classes of agents	**(1, 0, 0)**	**(0, 1 0)**	**(0, 0, 1)**
1	0.0008 (±0)	0.0008(±0)	0.0008 (±0)
2	19.1956 (±1.2)	61.8068 (±1.79)	83.9710 (±3.01)
3	18.6787 (±0.09)	48.2217 (±1.05)	61.6270 (±1.01)
4	18.2366 (±0.27)	48.8788 (±1.08)	63.0843 (±1.01)

Table 2: Spread statistics when only one influence factor is operative

- γ_j just represents the agent's intrinsic probability to change a mismatched bit. A low value of $\gamma_j < 0.5$ indicates an agent that does not easily change its mind, and a value higher than 0.5 an agent that is easier to influence.

While exploring networks with different types of agents, etc., is potentially a rich topic for research, in the present simulations, we set $\alpha_j = \beta_j = \gamma_j = 0.5\ \forall j$, and vary the relative weights, w_1, w_2 and w_3 for each influencer. In particular, we consider the following cases:

1. **Case 1 - One Influencer:** Here, one of the weights set to 1 and the other two to 0. This measures the effect of each factor's pure influence.

2. **Case 2 - Two Influencers:** In this case, two of the weights are set to 0.5 and the third to 0. Thus, the agent is equally influenced by two factors.

3. **Case 3 - Three Influencers:** Here, all weights are set to 1/3, so all factors have equal influence.

In each simulation run, a particlar agent of a specific type is chosen as the source and two of its opinion components are tagged with a color. As these bits influence other agents to change their opinions on these two components, those bits are also tagged. At the end of the simulation, any agent with one or two tagged bits is considered *influenced*. The percentage of agents influenced over a 150 step simulation is returned as the metric of influence. For each of the cases discussed below, 20 independent simulation runs were done with each type of agent as the source, so each case involved 80 separate runs.

5.2 Results

The results of our simulations for all three cases are discussed below.

Case 1: In the first set of simulations, one of the influence weights was set to 1 individually, with the others set to 0. The results are given in Table 2. Each entry represents the perceptage of the agents in the network reached after 150 time steps, with the values in parentheses indicating the standard deviation across the 20 trials.

Case 2: In the next set of simulations, two weights at a time were set to 0.5 and the third to 0, thus exploring the joint effect of two influencers at a time. The results are given in Table 3.

Classes of agents	(0, 0.5, 0.5)	(0.5, 0, 0.5)	(0.5, 0.5, 0)
1	0.0008 (±0)	0.0008(±0)	0.0008 (±0)
2	72.0206 (±2.04)	50.7890 (±1.07)	47.0286 (±2.01)
3	54.9748 (±1.13)	47.0176 (±1.27)	42.2600 (±1.36)
4	55.7595 (±1.18)	47.7735 (±1.12)	42.6811 (±1.42)

Table 3: Spread statistics when two influence factors are operative

Classes of agents	(1/3, 1/3, 1/3)
1	0.0008 (±0)
2	69.2181 (±2.11)
3	45.9241 (±1.91)
4	47.2163 (±1.01)

Table 4: Spread statistics when all influence factors are operative

Case 3: In these simulations, all weights were set to 1/3, thus giving each influence factor equal effect. The results were as shown in Table 4:

Several conclusions can be drawn from the results given above. First, it should be remembered that the efficacy of injecting opinions in one type of the agent or another depends on several things: 1) The inherent reach of the agent, i.e., how far can the agent itself disseminate opinions through direct connections; 2) The types of agents that it connects to, and their reach patterns; and 3) The chains of devotion linking agents along the tree of connections rooted at the injected agent. We currently do not have a detailed analysis of these factors, but the results given above reflect their combined influence.

First, we consider the effect of the influencing factors, f_1, f_2 and f_3, as determined by their weights. The results show that:

- Influence spreads least when the first factor – f_1, similarity of opinion – dominates. Given the model used, the mean initial similarity among any two agents is about 0.11, giving an influence factor $f_1 = 0.0584$. Thus when the weights are set as (1, 0, 0) (first table, column 1), any mismatched bit in a receiving agent has only a 5% chance of being influenced. It is also clear that, in this situation, the class of the injected agent has virtually no effect.

- Influence spreads most when the third factor – f_3, inherent receptiveness – dominates. This is because f_3 is set to a high value (0.5) for all agents, so a receiving agent has a 50-50 chance of changing a mismatched bit. In this case, the class of the injected agent does matter – probably due to differences in connectivity and clustering.

- When relative devotion is the only factor affecting influence (first table, column 2), the spread is between that seen in the other two cases, reflecting variation in relative devotion across the network. Here too, the type of agent matters, which is expected given the different distributions of relative devotion for each class.

Next, we consider the differences among agent types in terms of their efficacy as points of injection. The following effects are observed:

- Type 1 agents are of virtually no use in propagating opinions. This reflects their low connectivity, high clustering coefficients, and, bove all, their mostly positive relative devotion values – prcluding them for being influential even with the friends they do reach.

- On average, Type 2 agents are the most effective injection points for opinions. They have high connectivity and low clusterng coefficients, indicating the ability to spread opinions to many different parts of the network. With relative devotion values near zero, they also have significant influence (f_2 near 0.5) over their large circle of friends. Interestingly, the advantage of Type 2 agents is affected most strongly by the strength of the similarity factor (f_1) in the process. We speculate that this reflects the fact that most friends for Type 2 agents have devotion values near zero towards them, while friends of Type 3 and 4 agents have mostly positive values (as implied by the mostly negative devotion values of Type 3 and 4 agents themselves). Thus, Type 2 agents rely mainly on their reach and the inherent influenceability of their friends rather than devotion to convince others. When these factors are reduced, Type 2 agents lose almost all of their extra influence, while Type 3 and 4 agents can still rely on devition (factor f_2) to compensate somewhat.

- Type 3 and Type 4 agents have almost the same level of influence in all situations. Both classes have high degree and low clustering coefficients, so they can spread quite broadly. Additionally, their own mainly negative Δ values indicate their most of their friends have positive relative devotion towards them, so when $w_2 > 0$, their influence is boosted by this.

A full analysis of the effects of influencing factors and agent types requires a much more detailed investigation, both through more simulation cases and through analysis of network connectivity patterns. Results from such investigations will be reported in the future.

Acknowledgment This work was supported in part by a National Science Foundation CreativeIT grant to Ali Minai (IIS-0855714). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

6. REFERENCES

[1] L. Backstrom, D. Huttenlocher, and J. Kleinberg. Group Formation in Large Social Networks : Membership , Growth , and Evolution. *Science*, 2006.

[2] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–511, 1999.

[3] D. Crandall, D. Cosley, D. Huttenlocher, J. Kleinberg, and S. Suri. Feedback effects between similarity and social influence in online communities. In *In Proceedings of the 14th Annual ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2008.

[4] I. Derényi, G. Palla, and T. Vicsek. Clique Percolation in Random Networks. *Physical Review Letters*, 94(16):3–6, Apr. 2005.

[5] Facebook. Facebook statistics, Jan. 2011.

[6] M. Fraser and S. Dutta. The business advantages of social networking. *Finance and Management, the month magazine of the ICAEW's Finance and Management Faculty*, 168, 2009.

[7] R. Ghosh and K. Lerman. Predicting influential users in online social networks. In *In Proceedings of KDD workshop on Social Network Analysis (SNAKDD*, 2010.

[8] D. Kempe, J. Kleinberg, and . Tardos. Influential nodes in a diffusion model for social networks. In *In Proceedings of the 32nd International Conference on Automata, Languages, and Programming*, pages 1127–1138, 2005.

[9] D. Kempe, J. Kleinberg, and E. Tardos. Maximizing the spread of influence through a social network. In *In Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

[10] J. M. Kleinberg. Navigation in a small world. *Nature*, Aug. 2000.

[11] R. Kumar and A. Tomkins. Structure and Evolution of Online Social Networks. *Work*, pages 611–617, 2006.

[12] J. Leskovec, K. J. Lang, and M. W. Mahoney. Community Structure in Large Networks : Natural Cluster Sizes and the Absence of Large Well-Defined Clusters âĹŮ arXiv : 0810 . 1355v1 [cs . DS] 8 Oct 2008. 2008.

[13] T. M., A. Bodapati, and R. Bucklin. Determining influential users in internet social networks. *Journal of Marketing Research*, (643):2010.

[14] G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, pages 1–10, 2005.

[15] O. Pele and M. Werman. A linear time histogram metric for improved sift matching. In *ECCV*, 2008.

[16] O. Pele and M. Werman. Fast and robust Earth Mover's Distances. *2009 IEEE 12th International Conference on Computer Vision*, pages 460–467, Sept. 2009.

[17] F. Radicchi, C. Castellano, F. Cecconi, V. Loreto, and D. Parisi. Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 101(9):2658–63, Mar. 2004.

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[19] B. Simpkins, W. Sieck, P. Smart, and S. Mueller. Idea propagation in social networks: The role of cognitive advantage. In *1st ITA Workshop on Network-Enabled Cognition: The Contribution of Social and Technological Networks to Human Cognition*, July 2010.

[20] D. J. Watts and S. Strogatz. Collective dynamics of "small-world" networks. *Nature*, 393:440–442, 1998.

[21] P. Werbos. *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, 1974.

[22] C. Wilson, B. Boe, A. Sala, K. P. N. Puttaswamy, and B. Y. Zhao. User interactions in social networks and their implications. In *Proceedings of the 4th ACM European conference on Computer systems*, EuroSys '09, pages 205–218, New York, NY, USA, 2009. ACM.

The Emergence of Commitments and Cooperation

The Anh Han
Centro de Inteligência Artificial
Departamento de Informática,
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa,
2829-516 Caparical
Portugal
h.anh@campus.fct.unl.pt

Luís Moniz Pereira
Centro de Inteligência Artificial
Departamento de Informática,
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa,
2829-516 Caparical
Portugal
lmp@di.fct.unl.pt

Francisco C. Santos
Centro de Inteligência Artificial
Departamento de Informática,
Faculdade de Ciências e Tecnologia
Universidade Nova de Lisboa,
2829-516 Caparical
Portugal
fcsantos@fct.unl.pt

ABSTRACT

Agents make commitments towards others in order to influence others in a certain way, often by dismissing more profitable options. Most commitments depend on some incentive that is necessary to ensure that the action is in the agent's interest and thus, may be carried out to avoid eventual penalties. The capacity for using commitment strategies effectively is so important that natural selection may have shaped specialized capacities to make this possible. Evolutionary explanations for commitment, particularly its role in the evolution of cooperation, have been actively sought for and discussed in several fields, including Psychology and Philosophy. In this paper, using the tools of evolutionary game theory, we provide a new model showing that individuals tend to engage in commitments, which leads to the emergence of cooperation even without assuming repeated interactions. The model is characterized by two key parameters: the punishment cost of failing commitment imposed on either side of a commitment, and the cost of managing the commitment deal. Our analytical results and extensive computer simulations show that cooperation can emerge if the punishment cost is large enough compared to the management cost.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Multiagent systems

General Terms

Experimentation, Theory

Keywords

Evolution of Commitment, Evolution of Cooperation, Evolutionary Game Theory, Prisoner's Dilemma

1. INTRODUCTION

Over the last few years, several mechanisms have been pointed out to promote the emergence and maintenance of cooperation. From group and kin relations, memory and reputation based reciprocity mechanisms, to social diversity and context based reactions, grounded or not on incipient levels of cognition, there has been a large improvement on our capacity to understand the roots of animal and human cooperation [3, 2, 20, 27, 42, 45, 14, 15]. They are certainly hugely important, but are they sufficient? Or, as many have suggested [5, 19, 22], might there be other routes to social behavior that have been neglected? Certainly there are. Commitment, which amounts to expressing an intention rather than having it recognized, may stand as another route to cooperation, even in its simplest form, as we purport to show here.

Agents make commitments towards others when they give up options in order to influence others. Most commitments depend on some incentive that is necessary to ensure that the action is in the agent's interest and thus will be carried out [12], on pain of some heavy penalty. Committers also incur in a small cost when proposing or setting up a commitment so as to make it credible upfront to others, and entice these to accept to commit.

The capacity for using commitment strategies effectively is so important that natural selection may have shaped specialized signaling capacities to make this possible [49, 37, 41, 26, 8, 4]. And it is believed to have an incidence on the emergence of morality [38]. Assuming cooperation to be, at best, just the result of individuals' purely competitive strategies can make it conceptually unstable [31], most especially in non-iterated or history-free interactions. And it seems possible that the spread of simplistic notions, rooted in science, about the evolutionary origins of social relationships could foster a trend to make these relationships more conflicted, and society more brutal. An antidote is an evolutionary approach to behavior that incorporates a capacity for mutual commitment, shown advantageous for all concerned [26], even in non-iterated or memory-free settings.

Hence, our goal in this paper is to examine, through Evolutionary Game Theory (EGT) [20, 45], how the most simple of commitment strategies work, and how they can give rise to the emergence of cooperation. We shall do so in the setting of the non-iterated Prisoner's Dilemma (PD), a well-known game-theoretical framework to study the evolution of cooperation within populations of self-interested agents [2, 20, 27, 45] [1]. In an interaction, each player has two options,

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

[1]There are other social dilemmas, such as the Stag Hunt and the Chicken Game. [45], but the Prisoner's Dilemma is

cooperates (C) or defects (D), and defect is the dominant option – it is always better to defect in a one-shot interaction. Both players should choose to defect, while they would be better off by choosing to cooperate instead, thus leading to the destruction of social welfare and individuals' fitness.

In a nutshell, convincing others of one's credibility in a commitment proposal amounts to submit to options that change the incentives of the situation. These options, namely commitment cost and penalty for defaulting, can be expressed by the payoffs specified in a game. When opponent players observe meticulously such payoffs, and realize that compliance with a proposed commitment is in the proposing player's best interests, then, given any opponent player's open option to commit, these may change their expectations and behavior accordingly, and adopt as a result a strategy which either accepts commitment proposals or ignores them. In general, there are four main reasons to believe a commitment will be fulfilled [26]: i) a commitment can be self-reenforcing if it is secured by incentives intrinsic to the situation; ii) a commitment can be secured by external incentives controlled by third parties; iii) a commitment can be backed by a pledge of reputation; and iv) a commitment can be reinforced by internal emotional motives.

The first two types are secured in much the same way a loan is secured by a collateral. They objectively change the situation so that fulfillment becomes in the individual's best interests. The latter two types do not change the objective contingencies; they are subjective commitments in that they may involve a continued option of reneging, according to some or other stance extraneous to the game's given payoffs matrix.

In our EGT setting however, we will simply assume that a game's payoff matrix, concerning a set of strategies, summarily ingrains and expresses in its structure the impingement of all such contingencies. For instance, often a capacity for commitment allows individuals to act in ways that reap the benefits of image scoring through maintaining a reputation, or the access of others to a social history of prior interactions. In this study, for simplicity but also for exhibiting the purity and power of the commitment mechanism, we ignore the effect of repeated interactions [52], and of any reputation [29, 32] associated with particular individuals. We aim to show that the simplest of core commitment mechanisms can improve cooperation, and leave any other complications for the future, most promisingly how commitment can be combined with and reinforce other known mechanisms of cooperation. And perhaps surprisingly we can do so. Thus, no credibility of commitment will be taken into account [6] beyond that which is expressed in a game's payoff matrix. No reputation appraisal of the commitment proposer is made by its co-player, and no historical or social data is even available to do so. Each pairwise interaction is purely based on fixed individual strategies that might involve commitment or the lack thereof. Also, no "cheater or deceit detection" or "intention recognition" is in place [14, 15]. Nevertheless, systematic unconditional bluffing on the part of a player is a possible fixed feature of its strategy, in the sense that, from the start, the player does not intend to fulfill commitments.

It will be seen in our commitment model that players defaulting on their commitments, be they the proposing or the accepting party, will be subject to evolutionary disadvantage for a wide range of parameters.

We show that more elaborate commitment strategies are not strictly necessary for commitment to become evolutionarily advantageous. Neither an aptitude for higher cognition, nor for empathy, nor for mind reading are needed. These aptitudes would only be required for more sophisticated forms of commitment, scaffolded atop the core one. We will explain the evolution, in a population, of the capacity for a simple form of commitment as the result of otherwise being excluded from a group of committed promise abiding cooperators, in the sense that this strategy tends to invade the game playing population under rather general conditions.

The remainder of this paper is organized as follows. In Section 2, we discuss the relevant literature. In Section 3, our EGT commitment model and its methods are defined and explained. Forthwith, in Section 4, we proffer results obtained with the model, both analytic and via numeric and computer simulations. We conclude the paper with a discussion section on commitment and its EGT modeling.

2. RELATED WORK

Evolution of cooperation has been a central research topic of many fields, including Biology, Economics, Artificial Intelligence, Political Science and Psychology [2, 20, 27, 42, 45, 15, 21]. Several mechanisms responsible for promoting cooperative behavior have been recently identified (see surveys in [27, 45]). In these contexts, several aspects have been shown to play an important role in the emergence of cooperation. Differently, our model does not require any of those aspects, namely it does not assume kinship or ingroup relatedness of agents, nor repeated interactions or reputation consideration, nor concrete structures of population distribution. However, we envisage that the mechanism of commitment could reinforce the existing mechanisms of cooperations, e.g., easing the conditions for the emergence of cooperation therein. This will be the subject of the future work.

Evolutionary explanations of commitment, particularly its role in the evolution of cooperation, have been actively sought for and discussed in several fields, including Psychology and Philosophy [26, 12, 18, 6, 8, 4, 38]. But there are only a few computational models that show the evolutionary advantages of commitment in problems where cooperative acts are beneficial [49, 37, 41]. In addition, often models rely on repeated interactions or long-term relationships [8, 4], alike the conditions where Triver's direct reciprocity [52] may play a role. Here we provide an analytic model in the framework of evolutionary game theory showing that, with the availability of the mechanism of commitment, cooperation can emerge even without assuming repeated interactions.

Last but not least, it is undoubtedly important to mention the extensive literature of AI and Multi-agent System research on commitment, e.g., [43, 54, 18, 6, 53, 16, 7]. The main concern therein is how to formalize different aspects of commitment and how a commitment mechanism can be implemented in multi-agent interactions to enhance them (e.g. for improved collaborative problem solving [54]), especially in the context of game theory. In contradistinction, our concern is in the nature of an evolutionary explanation of commitment, particularly how it can promote the emergence of cooperation.

known to represent one of the most difficult or fierce environments for cooperation to emerge.

3. MODELS AND METHODS

3.1 Model

Let us consider a commitment variant of the Prisoner's Dilemma game in which a new type of cooperator (denoted by COM_C) that, before each interaction, asks the co-player whether it commits to cooperate. If the co-player does not so commit, there is no interaction. Both players get 0. Otherwise, if the co-player commits, they then go on to play with each other in the present interaction. If the co-player keeps to its commitment, both players obtain the reward payoff, R [2] . Otherwise (if the co-player fails its commitment), the proposing or focal player obtains the sucker payoff, S, and its co-player obtains the temptation payoff, T. However, the one that fails the commitment will suffer a penalty cost, and its non-defaulting co-player gains a compensation for the potential loss due to its default of fulfilling the commitment. For simplicity, we assume that these two amounts (penalty and compensation) are equal, being denoted by δ. The penalty cost can be a real monetary one, e.g., in the form of prior debit (e.g., in the case of accommodation rental) or of a subsequent punishment cost (e.g., commitment was performed in terms of a legal contract, and one who fails commitment must pay a cost to compensate for the other), or an imaginary abstract value, e.g., public spread of good/bad reputation (bad reputation for the one that fails, and sympathy for the other), or even an emotional suffering [26, 12, 18, 38]. How this cost is set up depends on the types of commitment at work, or the reason for which the commitment is believed to be fulfilled (see Introduction), which topic is beyond the scope of this paper. However, various techniques can be seen in [43, 18].

Two players that defect in an interaction obtain the punishment payoff, P [3] . As usual, for the Prisoner's Dilemma, the payoff entries satisfy the ordering, $T > R > P > S$, whereas the four possible outcomes can be written down as a payoff matrix

$$\begin{array}{c} \\ C \\ D \end{array} \begin{array}{c} \begin{array}{cc} C & D \end{array} \\ \begin{pmatrix} R,R & S,T \\ T,S & P,P \end{pmatrix} \end{array}$$

For setting up a commitment, the proposer must pay a small management cost, ϵ. The cost of proposing and setting up the commitment might be high, but it is reasonable to assume that this cost is quite small compared to the mutual benefit of a cooperation strategy guaranteeing commitment, $\epsilon << R$.

Given the nature of a situation expressed in terms of payoff entries, one can naturally expect that if a proposed punishment cost, δ, is high enough compared to the cost of managing the commitment, ϵ – to convince and guarantee that cooperation is in the proposer's interest and also drive away potential exploiters – cooperation can emerge, even in the fierce environment of the Prisoner's Dilemma. This penalty and management relation is subject to detailed study below, both analytically and by means of computer simulations.

We consider a finite population of a constant size, consisting of four strategies: COM_C (as described above), C (always cooperates, without proposing to commit), D (always defects, and does not commit when being asked to), and D_COM (always defects, though commits when being asked to). Here, we assume that cooperators, including COM_C and C players, always commit whenever being asked to since they are better off to do so, as cooperation is their default choice, and reasonable commitment deals only are proposed. Hence, for the sake of exposition, the two other (unreasonable) strategies, those of cooperators that refuse to commit and of defectors that propose commitment, are omitted here (they would become eliminated anyway). The former is dominated by the pure cooperator strategy, C, while the latter is by the pure defector strategy, D.

In each round, two random players are chosen from the population for an interaction. For the row player, the (average) payoff matrix reads

$$\begin{array}{c} \\ COMC \\ C \\ D \\ DCOM \end{array} \begin{array}{c} \begin{array}{cccc} COMC & C & D & DCOM \end{array} \\ \begin{pmatrix} R-\epsilon/2 & R-\epsilon & -\epsilon & S+\delta-\epsilon \\ R & R & S & S \\ 0 & T & P & P \\ T-\delta & T & P & P \end{pmatrix} \end{array} \quad (1)$$

Note that when a COM_C interacts with another COM_C, only one of them pays the cost of having proposed commitment, ϵ (e.g., the arbitrary one that proposes). Therefore, the average payoff of a COM_C in playing with another COM_C is, $R - \epsilon/2$.

3.2 Methods

Our analysis is based on evolutionary game theory methods for finite populations [28, 23]. In the context of evolutionary game theory, the individuals' or agents' payoff represents their *fitness* or social *success*. The dynamics of strategy change in a population is governed by social learning, that is, the most successful agents will tend to be imitated by the others. There are many ways to model social learning [20, 45, 36]. Adopting one of the most frequently used ones, we shall consider the so-called pairwise comparison rule [51], which assumes that an agent A with fitness f_A adopts the strategy of another agent B with fitness f_B with probability given by

$$\frac{1}{1+e^{-\beta(f_B-f_A)}},$$

where β controls the 'imitation strength', i.e., how strongly the agents are basing the decision to imitate on fitness comparisons. For $\beta = 0$, we obtain the limit of neutral drift – the imitation decision is random. For large β, imitation becomes increasingly deterministic.

In the absence of mutations, the end states of evolution are inevitably monomorphic: once such a state is reached, imitation cannot produce change. We thus further assume that, with a certain mutation probability $\mu > 0$ (also dubbed the exploration rate [50]), an agent switches randomly to a different strategy without imitating another agent. The resulting Markov Chain has a stationary distribution, which characterizes the average time the population spends in each of these monomorphic end states. Yet, for arbitrary exploration rates and number of strategies, stationary distribu-

[2]Note that here we do not yet take into account execution noise (see, e.g., [45, 32]), i.e. the agents might mis-implement their intended choice, from cooperate to defect or vice versa. Thus, COM_C will never mis-implement the intended commitment choice.

[3]For the sake of a clear representation, in our analysis we adopt $P = 0$ [40, 45] (as in the Donation game), even if the more general case can be analyzed in the same manner and portray similar results to the ones presented below.

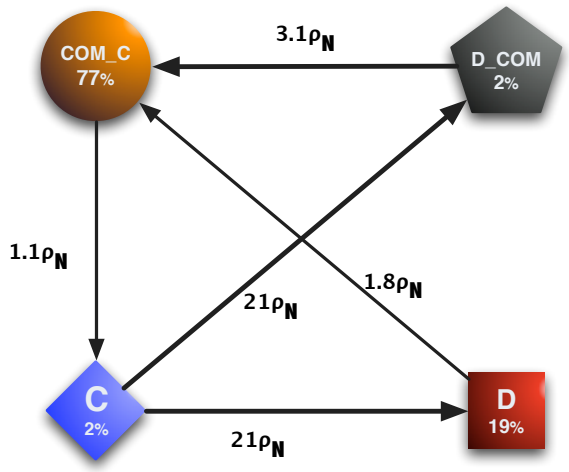


Figure 1: Stationary distribution and fixation probabilities. The population spends most of the time in the homogenous state of COM_C. The black arrows stand for the transitions that are rather stronger than neutral. The strongest transitions are from C to D and D_COM, and the slowest one is from COM_C to C. There are rather strong transitions from D and D_COM to COM_C. Parameters: $T = 2,\ R = 1,\ P = 0,\ S = -1; \delta = 4;\ \epsilon = 0.05$**; imitation strength,** $\beta = 1$**; population size,** $N = 100$**;** $\mu_N = 1/N$ **denotes the neutral fixation probability.**

tions are often cumbersome to compute [17, 46, 39].

Fortunately, in the case of small exploration or mutation rates, analytical computation of this stationary distribution can be conveniently computed [11, 23, 17, 41]. The small exploration rates guarantee that, any newly occurred mutant in a homogeneous population will fixate or become extinct long before the occurrence of another mutation. Hence, the population will always consist of at most two strategies. This allows one to describe the evolutionary dynamics of our population in terms of a reduced Markov Chain, whose size is equal to the number of strategies being considered (which is 4 in our case), and each state represents a possible monomorphic end state of the population associated with a one of the strategies. The transitions between states are defined by the fixation probabilities of a single mutant of one strategy in a homogeneous population of individuals adopting another strategy (see Figure 1 for better understanding).

$$T^{\pm}(k) = \frac{N-k}{N}\frac{k}{N}\frac{1}{1+e^{\mp\beta[\Pi_A(k)-\Pi_B(k)]}} \tag{2}$$

More precisely, let N be the size of the population. Suppose there are at most two strategies in the population, say, k agents using strategy A ($0 \leq k \leq N$) and $(N-k)$ agents using strategies B. Thus, the (average) payoff of the agent that uses A and B can be written as follows, respectively,

$$\begin{aligned}\Pi_A(k) &= \frac{(k-1)\pi_{A,A} + (N-k)\pi_{A,B}}{N-1} \\ \Pi_B(k) &= \frac{k\pi_{B,A} + (N-k-1)\pi_{B,B}}{N-1}\end{aligned} \tag{3}$$

where $\pi_{X,Y}$ stands for the payoff an agent using strategy X obtained in an interaction with another agent using strategy Y, given in the payoff matrix (1).

Now, the probability to change the number k of agents using strategy A by $\pm$ one in each time step can be written as

The fixation probability of a single mutant with a strategy A in a population of $(N-1)$ agents using B is given by [51, 25, 11, 23, 17]

$$\rho_{B,A} = \frac{1}{1+\sum_{i=1}^{N-1}\prod_{j=1}^{i}\frac{T^-(j)}{T^+(j)}} \tag{4}$$

In the limit of neutral selection ($\beta = 0$), $T^-(j) = T^+(j)\ \forall j$. Thus, $\rho_{B,A} = 1/N$. Considering a set $\{1, ..., q\}$ of different strategies, these fixation probabilities determine a transition matrix $M = \{T_{ij}\}_{i,j=1}^{q}$, with $T_{ij,j\neq i} = \rho_{ji}/(q-1)$ and $T_{ii} = 1 - \sum_{j=1,j\neq i}^{q} T_{ij}$, of a Markov Chain. The normalized eigenvector associated with the eigenvalue 1 of the transposed of M provides the stationary distribution described above [25, 11, 23, 17], describing the relative time the population spends adopting each of the strategies.

Now let us recall some important analytic measures which will be used in our analytical study. In a pair-wise comparison of strategy A with strategy B, we say that A is advantageous (against B) if an A mutant has a fixation probability in a population of agents using B greater than that of the neutral selection (which equals the inverse of population size, $1/N$) [28, 27, 45]. Interestingly, it was shown that this condition holds if

$$(N-2)\pi_{A,A} + (2N-1)\pi_{A,B} > (N+1)\pi_{B,A} + (2N-4)\pi_{B,B} \tag{5}$$

which, in the limit of large N, is simplified to

$$\pi_{A,A} + 2\pi_{A,B} > \pi_{B,A} + 2\pi_{B,B} \tag{6}$$

Another important measure to compare the two strategies A and B is which direction the transition is stronger or more probable, an A mutant fixating in a population of agents using B or a B mutant fixating in the population of agents using A. It can be shown that the former is stronger if [24, 45]

$$(N-2)\pi_{A,A} + N\pi_{A,B} > (N-2)\pi_{B,A} + N\pi_{B,B} \tag{7}$$

which, in the limit of large N, is simplified to

$$\pi_{A,A} + \pi_{A,B} > \pi_{B,A} + \pi_{B,B} \tag{8}$$

4. RESULTS

We compute the fixation probabilities and stationary distribution numerically for small mutation or exploration rates (see Methods). The population spends most of the time in the homogeneous state where all individuals utilize the commitment strategy (Figure 1).

In general, amongst the monomorphic states of the population, the strongest transitions are from C to D and C to D_COM. The difference of a small cost of proposing commitment, ϵ, between COM_C and C, leads to a near-neutral transition from COM_C to C. The more intricate transitions are between COM_C and D or D_COM, which are the central part of our analysis.

Between D and COM_C, for $\epsilon << R$, COM_C is advantageous. Namely, by a pairwise comparisons of COM_C and D

[28, 27] that condition always holds if (in the limit of large N, see Eq. (6))

$$\epsilon < \frac{2R}{5} \tag{9}$$

This inequality also guarantees that, for a population of size $N > 4$, the more probable transition is from D to COM_C, i.e., satisfying that [24, 45] (see Eq. 8)

$$(N-2)(R-\frac{\epsilon}{2}) - N\epsilon > 0 \tag{10}$$

Similarly, for big enough δ, COM_C is advantageous against D_COM; namely, if

$$\delta > \frac{T-R-2S}{3} + \frac{5\epsilon}{6} \tag{11}$$

It guarantees that the transition of D_COM to COM_C is more probable than the opposite if

$$(N-2)(R-\frac{\epsilon}{2}) + N(S+\delta-\epsilon) > (N-2)(T-\delta) \tag{12}$$

which holds if

$$\delta > \frac{N}{2N-2}(T-R-S+\frac{3\epsilon}{2}) \tag{13}$$

For large enough N, it is simplified to

$$\delta > \frac{T-R-S}{2} + \frac{3\epsilon}{4} \tag{14}$$

Hence, for

$$\delta > \max\{\frac{T-R-S}{2} + \frac{3\epsilon}{4}, \frac{T-R-2S}{3} + \frac{5\epsilon}{6}\} \tag{15}$$

the transition of D_COM to COM_C is the more probable one, as well as greater than neutral.

Taking together with the fact that the transition of COM_C to C is near neutral, one can expect that if the two parameters δ and ϵ satisfy the inequalities (9) and (15), COM_C will prevail – the population will spend most of the time in its homogenous state. This expectation is supported by the numerical results in Figures 2 and 3. For a given payoff matrix of the PD, for strong enough punishment cost of failing commitment, δ, and small enough cost of setting up the commitment, ϵ, the population spends most of the time in the homogeneous state of COM_C (Figure 2). In addition, this result is also flexible with respect to the payoff values of the PD (Figure 3). For the sake of a clear representation of the result, we use in this numerical experiment the Donation game [46] – a special case of PD – where $T = b,\ R = b-c,\ P = 0,\ S = -c$, satisfying that $b > c > 0$; b and c stand for "benefit" and "cost" of cooperation, respectively. It shows that, for given δ and ϵ, for large enough b/c, the population spends most of the time in the homogeneous state of COM_C.

So far, our analytic and numerical results were obtained in the limit of small mutation rates. Next, by extensive computer simulations, we show that this remarkable performance of the commitment strategy COM_C is flexible with respect to mutation rates (Figure 4). Namely, for all the mutation rates up to 0.1, the population always spends most of the time in the homogenous state of COM_C. It also noteworthy, that our analytic results for small imitation strengths and under the extremes of low and high mutation or exploration rates — based on the methods described in [1] — comply with this simulation results.

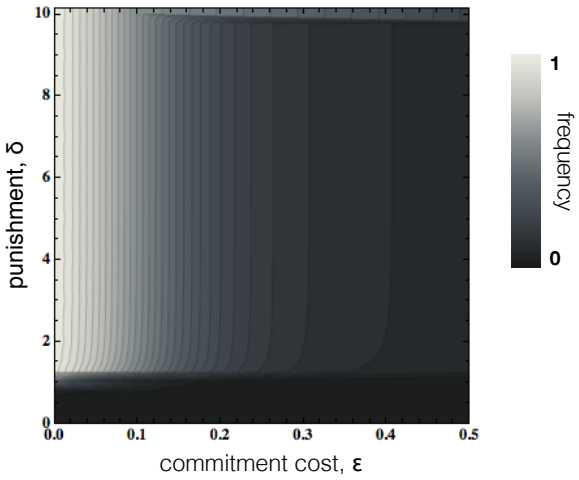


Figure 2: Contour plot of the frequency of COM_C as a function of ϵ and δ. In a population of COM_C, COM_D, C, and D individuals, for a wide range of ϵ and δ, the population spends most of the time in the homogeneous state of COM_C. The smaller the cost of proposing commitment, ϵ, and the greater the punishment cost of failing commitment, δ, the greater the frequency of COM_C. The payoffs being used are, $T = 2,\ R = 1,\ P = 0,\ S = -1$; imitation strength, $\beta = 1$; population size, $N = 100$.

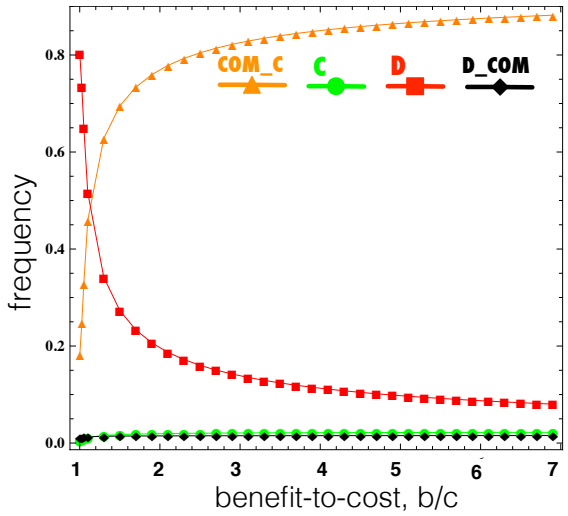


Figure 3: Frequency of each strategy as a function of benefit-to-cost ratio, b/c, for Donation game ($T = b,\ R = b-c,\ P = 0,\ S = -c$, with $b \geq c$). In a population of COM_C, COM_D, C, and D individuals, for a large enough benefit-to-cost ratio, the population spends most of the time in the homogeneous state of COM_C, while D prevails when this ratio is very small. Parameters: $\delta = 4; \epsilon = 0.05$; imitation strength, $\beta = 1$; population size, $N = 100$.

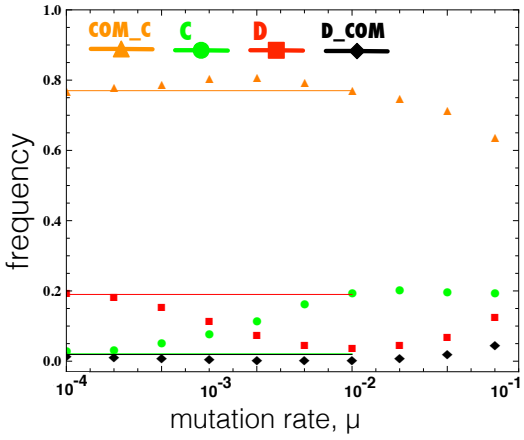


Figure 4: Frequency as a function of mutation rates. Symbols indicate results from computer simulations (averaged over 10^9 update steps), and solid lines show the numerical approximation results for small mutation rates. The population consists of COM_C, COM_D, C, and D individuals. COM_C always dominates for all the mutation rates up to 0.1. Parameter values (the same as in Figure 1): $T = 2, R = 1, P = 0, S = -1; \delta = 4; \epsilon = 0.05$; imitation strength, $\beta = 1.0$; population size, $N = 100$. The simulation results are obtained by averaging 40 runs, and the initial population is equally likely to be in one of the homogenous states.

All in all, our study exhibits that, in spite of the absence of repeated interactions, reputation effect, network reciprocity, as well as group and kin selection, the strategy of commitment proposal may enable the emergence of cooperation. By imposing a high cost for failing a commitment, when compared to the cost of setting up or managing the commitment deal, the commitment cooperative agents COM_C can get rid of the fake committers (D_COM) as well as avoid being exploited by the pure defectors (D), while playing approximately equally well against the pure cooperators (C). The results of this study suggest that our specialized capacity for commitment, which might have been shaped by natural selection [26] consists in a capacity for managing to impose a high cost of punishment, whether it is monetary or of abstract emotional or reputation value, with a relatively small cost.

We note that there is a significant difference between our commitment model and the works on costly punishment [30, 35, 9, 17, 10]. A commitment deal must be agreed by both sides of it in advance, thereby giving credibility and justification to punish any defaulting player. In addition, the prior agreement gives rise to compensation—the amount of which, in some cases, is agreed explicitly in advance—to the non-defaulting player. This compensation for the non-defaulting player is the significant difference that makes successful those players using the commitment strategy, while those using the costly punishment strategy have only a narrow margin of efficiency [30]; does not stand out as a winning strategy [9]; nor does it promote cooperation at all when taking into account antisocial punishment [35]. The compensation might bring benefit to the commitment strategists once an appropriate deal would be arranged.

This suggests that although costly punishment, whether it is social or antisocial, might not promote the evolution of cooperation, what we call 'justified' punishment, which is warranted by an appropriate commitment deal, does. This kind of punishment might not be costly at all, and can even bring net benefit to its upholder, hence leading to the emergence of cooperation.

5. DISCUSSIONS

Within the general game theory concept of commitment, several distinctions can help separate different subtypes. In particular, some commitments are upfront promises of a next move that can help, while others are upfront threats of a subsequent move that can harm. Commitments can be conditional or unconditional. Threats are usually attempts to influence another person's next move by stating a conditional subsequent move, and that's how we may envisage them. Promises are more likely to be unconditional, and that's how we may conceive of them, though more generally they can be conditional on the other fulfilling a matching promise. Concerning this, we note a difference between a commitment and a convention. A convention is a means for monitoring a commitment: it specifies under what circumstances a commitment can be abandoned and how an agent should behave both locally and towards others when one of these conditions arises [54]. Commitments can also be just towards oneself, taking into account the evolution of possible futures afforded by actions and events, and the individual's prior and post preferences, in what might be classically seen as a game against nature.

In [34, 33], three different types of individual commitment – hard, revocable, and momentary – are studied in such an evolution context. Let us recall that commitment, in the context of game theory, is a device or mechanism to decide the outcome with the other party [43]. Schelling distinguishes between commitment pure and simple and commitment that takes the form of a threat. What he calls "ordinary" commitment corresponds, in game theory, to the making of an opening announcement in a sequential play, which we dub preemptive, just before both players make their actual move. To constitute a preemption, a player's announcement action must be irrevocable, that is a promise that is assuredly kept. Preemptive commitment is not necessarily profitable, because it hinges on the opponent's actual move. Schelling however does not assume the other type of commitment as a "threat", which pertains to the a player's move in reaction to the opponent's move. Threats, being conditional, may be of the "if-then-else" form, and can thus combine a threat and a promise, the latter albeit implicit whenever there are just two possible moves. We prefer instead to label "reactive" such so-called threat commitments. In the game context, these occur when the player with the last move irrevocably pledges to respond, in a specified but contingent way, to the opponent's prior choice [19].

In a nutshell, some players can be "preemptive" committers – those that always propose and always accept proposed commitments–, others may be "reactive" committers – those that always make a "reactive" statement and comply with the implicit requests in such statements–, while other players, though accepting to commit nevertheless default on their commitment, and even others simply omit and ignore preemptive or reactive commitments in their strategies –

they might for instance be persistent defectors or persistent cooperators as we have seen, or, for that matter, follow any other strategy ignorant of commitment. Moreover, in iterated games, commitments can concern future rounds and not just the present one.

We purport to have shown that a simple commitment abiding cooperative strategy can be evolutionarily advantageous even in a non-iterated game setting. But much remains to be explored. In the more general setting and to avoid confusion, it can be helpful to distinguish, even if only conceptually, between "execution moves" and "pre-play moves" [19]. The terms first move and last move then always refer exclusively to execution moves – the choices that actually generate the payoffs. In contrast, commitments come earlier with respect to execution moves: they are pre-play moves. A preemptive commitment is a pre-play move that allows the player making it to take the first execution move. A reactive commitment, although also a pre-play move, can be made only by the player who has the last execution move. In either case, by giving up on his or her choice through committing, the commitment player leaves the opponent with "the last clear chance to decide the outcome" [43].

In our present game setting, however, there was no need to make the distinction between the first and the second to play, because each possible player strategy move is exhibited and fixed from the start, as expressed and ingrained in the payoff matrix. By so introducing the several committed unconditional move strategies – though the payoff is of course conditional on the opponent's move–, we can emulate what would happen in a round if a move sequence actually existed. Put briefly, our commitment model is of the simplest kind and, moreover, it is brought to bear solely on the very next move fold of a pair of players, with no history available on prior commitments. Nevertheless, it captures core features of commitment, namely the high cost of defaulting to discourage false commitment, and thus make it plausible, and a comparatively small but non-zero cost of commitment proposal to lend it initial credibility. On top of this core model more elaborate models affording commitment can subsequently be rooted, including those involving delayed deceit..

What's more, commitment (or intention manifestation) and intention recognition, are but two sides of a coin really, and their future joint study in the EGT setting is all but unavoidable. It has become increasingly obvious that maximizing reproductive success often requires keeping promises and fulfilling threats, even when that requires in turn sacrifices regarding individual short-term interests. That natural selection has shaped special mental capacities to make this possible seems likely, including a capacity for commitment [26] and for intention recognition [14, 15]. The commitment stance goes yet further, and many aspects of human groups seem shaped by effects of commitments and intention recognition, namely group boundaries, initiation rituals, ideologies, and signals of loyalty to the group [47, 48, 49]. Conversely, many aspects of groups seem to exist largely to facilitate commitment to cooperate and to limit the utility of coercive threats.

The generalized ability for commitment to support cooperative interaction is an important aspect of plasticity in human behavior, and humans support their deal-making in lots of ways. The law is full of instances of people using techniques of commitment to establish the honesty of their intentions, namely through a variety of contracts [13]. Institutions themselves are supported on committal contracts, and the law of the land proffers methods for constituting and of accountability of social institutions [44].

We believe that studies of commitment will benefit greatly from rigorous models that allow for their analytical study and computer simulation, and in particular within the fold of EGT for the better to examine the emergence of complex social behavior.

6. ACKNOWLEDGMENT

HTA and FCS acknowledge the support from FCT-Portugal (grant SFRH/BD/62373/2009 and R&D project PTDC/FIS/101248/2008, respectively).

7. REFERENCES

[1] T. Antal, A. Traulsen, H. Ohtsuki, C. E. Tarnita, and M. A. Nowak. Mutation-selection equilibrium in games with multiple strategies. *J. Theor. Biol.*, 258:614–622, 2009.

[2] R. Axelrod. *The Evolution of Cooperation*. Basic Books, ISBN 0-465-02122-2, 1984.

[3] R. Axelrod and W. Hamilton. The evolution of cooperation. *Science*, 211:1390–1396, 1981.

[4] I. Back and A. Flache. The Adaptive Rationality of Interpersonal Commitment. *Rationality and Society*, 20(1):65–83, 2008.

[5] C. Boehm. The natural selection of altruistic traits. *Human Nature*, 10(3):205–252, 1999.

[6] C. Castelfranchi and R. Falcone. *Trust Theory: A Socio-Cognitive and Computational Model (Wiley Series in Agent Technology)*. Wiley, 2010.

[7] A. K. Chopra and M. P. Singh. Multiagent commitment alignment. In *Proceedings of the 8th International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, pages 937–944, 2009.

[8] de Vos, R. Smaniotto, and D. Elsas. Reciprocal altruism under conditions of partner selection. *Rationality and Society*, 13(2):139–183, 2001.

[9] A. Dreber, D. G. Rand, D. Fudenberg, and M. A. Nowak. Winners don't punish. *Nature*, 452(7185):348–351, 2008.

[10] E. Fehr and S. Gachter. Altruistic punishment in humans. *Nature*, 415:137–140, 2002.

[11] D. Fudenberg and L. A. Imhof. Imitation processes with small mutations. *Journal of Economic Theory*, 131:251–262, 2005.

[12] H. Gintis. Beyond selfishness in modeling human behavior. In R. M. Nesse, editor, *Evolution and the capacity for commitment*. New York: Russell Sage, 2001.

[13] O. R. Goodenough. Law and the biology of commitment. In R. M. Nesse, editor, *Evolution and the capacity for commitment*, pages 262–291. New York: Russell Sage, 2001.

[14] T. A. Han, L. M. Pereira, and F. C. Santos. Intention recognition promotes the emergence of cooperation. *Adaptive Behavior*, 19(3):264–279, 2011.

[15] T. A. Han, L. M. Pereira, and F. C. Santos. The role of intention recognition in the evolution of cooperative

behavior. In *Proceedings of the 22nd international joint conference on Artificial intelligence (IJCAI'2011)*, pages 1684–1689, 2011.

[16] P. Harrenstein, F. Brandt, and F. Fischer. Commitment and extortion. In *The 6th international joint conference on Autonomous agents and MultiAgent systems*, AAMAS '07. ACM, 2007.

[17] C. Hauert, A. Traulsen, H. Brandt, M. A. Nowak, and K. Sigmund. Via freedom to coercion: The emergence of costly punishment. *Science*, 316:1905–1907, 2007.

[18] J. Hirshleifer. Game-theoretic interpretations of commitment. In R. M. Nesse, editor, *Evolution and the capacity for commitment*, pages 77–93. New York: Russell Sage, 2001.

[19] J. Hirshleiffer. There are many evolutionary pathways to cooperation. *Journal of Bioeconomics*, (1):73–93, 1999.

[20] J. Hofbauer and K. Sigmund. *Evolutionary Games and Population Dynamics*. Cambridge U. P., 1998.

[21] L.-M. Hofmann, N. Chakraborty, and K. Sycara. The evolution of cooperation in self-interested agent societies: a critical study. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '11, pages 685–692, 2011.

[22] N. K. Humphrey. Varieties of altruism. *Social Research*, (64):199–209, 1999.

[23] L. A. Imhof, D. Fudenberg, and M. A. Nowak. Evolutionary cycles of cooperation and defection. *Proc. Natl. Acad. Sci. USA*, 102:10797–10800, 2005.

[24] M. Kandori, G. J. Mailath, and R. Rob. Learning, mutation, and long run equilibria in games. *Econometrica*, 61:29–56, 1993.

[25] S. Karlin and H. E. Taylor. *A First Course in Stochastic Processes*. Academic Press, New York, 1975.

[26] R. M. Nesse. Natural selection and the capacity for subjective commitment. In R. M. Nesse, editor, *Evolution and the capacity for commitment*, pages 1–44. New York: Russell Sage, 2001.

[27] M. A. Nowak. Five rules for the evolution of cooperation. *Science*, 314(5805):1560, 2006. DOI: 10.1126/science.1133755.

[28] M. A. Nowak, A. Sasaki, C. Taylor, and D. Fudenberg. Emergence of cooperation and evolutionary stability in finite populations. *Nature*, 428:646–650, 2004.

[29] M. A. Nowak and K. Sigmund. Evolution of indirect reciprocity. *Nature.*, 437(7063):1291–1298, 2005.

[30] H. Ohtsuki, Y. Iwasa, and M. A. Nowak. Indirect reciprocity provides only a narrow margin of efficiency for costly punishment. *Nature*, 457(7601):79–82, 2009.

[31] S. Oyama. *Evolution's Eye: A Systems View of the Biology-Culture Divide*. Durham, N.C.: Duke University Press., 2000.

[32] J. M. Pacheco, F. C. Santos, and F. A. C. C. Chalub. Stern-judging: A simple, successful norm which promotes cooperation under indirect reciprocity. *PLoS Comput Biol*, 2:12:e178, 2006.

[33] L. M. Pereira and T. A. Han. Evolution prospection. In *Proceedings of International Symposium on Intelligent Decision Technologies (KES-IDT'09)*, pages 51–63. Springer Studies in Computational Intelligence 199, 2009.

[34] L. M. Pereira and T. A. Han. Evolution prospection in decision making. *Intelligent Decision Technologies*, 3(3):157–171, 2009.

[35] D. G. Rand and M. A. Nowak. The evolution of antisocial punishment in optional public goods games. *Nature Communications*, 2:434, 2011.

[36] L. Rendell, R. Boyd, D. Cownden, M. Enquist, K. Eriksson, M. W. Feldman, L. Fogarty, S. Ghirlanda, T. Lillicrap, and K. N. Laland. Why copy others? insights from the social learning strategies tournament. *Science*, 328(5975):208–213, 2010.

[37] A. Robson. Efficiency in evolutionary games: Darwin, nash, and the secret handshake. *J Theo Biol*, 144:379–396, 1990.

[38] M. Ruse. Morality and commitment. In R. M. Nesse, editor, *Evolution and the capacity for commitment*, pages 221–236. New York: Russell Sage, 2001.

[39] F. C. Santos and J. M. Pacheco. Risk of collective failure provides an escape from the tragedy of the commons. *Proc Natl Acad Sci USA*, 108:10421–5, 2011.

[40] F. C. Santos, J. M. Pacheco, and T. Lenaerts. Evolutionary dynamics of social dilemmas in structured heterogeneous populations. *Proc. Natl. Acad. Sci. USA*, 103:3490–3494, 2006.

[41] F. C. Santos, J. M. Pacheco, and B. Skyrms. Co-evolution of pre-play signaling and cooperation. *J Theo Biol*, 274:30–35, 2011.

[42] F. C. Santos, M. D. Santos, and J. M. Pacheco. Social diversity promotes the emergence of cooperation in public goods games. *Nature*, 454:214–216, 2008.

[43] T. C. Schelling. *The strategy of conflict*. London: Oxford University Press, 1990.

[44] J. R. Searle. *Making the Social World: The Structure of Human Civilization*. Oxford University Press, 2010.

[45] K. Sigmund. *The Calculus of Selfishness*. Princeton U. Press, 2010.

[46] K. Sigmund, H. D. Silva, A. Traulsen, and C. Hauert. Social learning promotes institutions for governing the commons. *Nature*, 466:7308, 2010.

[47] B. Skyrms. *Evolution of the Social Contract*. Cambridge University Press, 1996.

[48] B. Skyrms. *The Stag Hunt and the Evolution of Social Structure*. Cambridge University Press, 2003.

[49] B. Skyrms. *Signals: Evolution, Learning, and Information*. Oxford University Press, 2010.

[50] A. Traulsen, C. Hauert, H. De Silva, M. A. Nowak, and K. Sigmund. Exploration dynamics in evolutionary games. *Proc. Natl. Acad. Sci. USA*, 106(3):709–712, 2009.

[51] A. Traulsen, M. A. Nowak, and J. M. Pacheco. Stochastic dynamics of invasion and fixation. *Phys. Rev. E*, 74:11909, 2006.

[52] R. L. Trivers. The evolution of reciprocal altruism. *The Quarterly Review of Biology*, 46:35–57, 1971.

[53] M. Winikoff. Implementing commitment-based interactions. In *The 6th international joint conference on Autonomous agents and multiagent systems*, AAMAS '07, pages 868–875. ACM, 2007.

[54] M. Wooldridge and N. R. Jennings. The cooperative problem-solving process. In *Journal of Logic and Computation*, pages 403–417, 1999.

Author Index

Abdallah, Sherief, 1381
Adams, Julie, 569, 593
Agmon, Noa, 341, 1251
Ågotnes, Thomas, 1099
Alan, Perotti, 1023
Alberola, Juan M., 1379
Alberti, Marco, 1425
Albrecht, Stefano, 349
Alcântara, João, 501
Aldewereld, Huib, 1371, 1421
Alechina, Natasha, 1057, 1099, 1309
Alers, Sjriek, 1475
Alferes, José, 1423
Alford, Ron, 981
Allahverdyan, Armen, 1391
Almajano, Pablo, 1483
Alvarado, Oscar, 1493
Amarante, Maicon, 1351
Amor, Mercedes, 1427
An, Bo, 13, 863, 1307
Anand, Sarabjot Singh, 1367
Andrighetto, Giulia, 1189
Anshelevich, Elliot, 1321
Antos, Dimitrios, 55
Aravamudhan, Ajay Srinivasan, 1227
Argente, Estefanía, 1419
Artstein, Ron, 63
Athakravi, Duangtida, 1369
Atkinson, Katie, 1171
Aumann, Yonatan, 1293
Ayala, Inmaculada, 1427
Aylett, Ruth, 1197, 1457
Azar, Yossi, 897
Azaria, Amos, 459
Aziz, Haris, 585, 763, 1311

Bachrach, Yoram, 535
Bai, Aijun, 1215
Bai, Quan, 1459
Baier, Jorge, 997
Baldwin, Craig, 13
Banerjee, Bikramjit, 1441
Baptista, Márcia, 1175
Barlow, Gregory J., 1271
Barrera, Francisco, 129
Barrett, Samuel, 129, 357
Baumeister, Dorothea, 577
Bazzan, Ana, 1351, 1389, 1395
Becerik-Gerber, Burcin, 21, 1455
Bekris, Kostas, 247
Bench-Capon, Trevor, 1171
Benenson, Itzhak, 1453
Bergenti, Federico, 1435
Berzan, Constantin, 189
Bessiere, Christian, 1263
Beygelzimer, Alina, 1317
Bianchi, Reinaldo, 1395
Bída, Michal, 1469, 1477
Biec, Santiago, 1481
Bill-Clark, Luis, 1213
Birattari, Mauro, 139
Bistaffa, Filippo, 1461
Bloembergen, Daan, 1393, 1475
Blokzijl-Zanker, Michiel, 1207
Boella, Guido, 1023
Bölöni, Ladislau, 1345
Bonjean, Noélie, 1065
Bonzon, Elise, 1413
Booth, Richard, 493
Bošanský, Branislav, 905, 1301, 1473
Botia, Juan, 307
Botti, Vicent, 1355, 1377, 1419, 1493
Bou Ammar, Haitham, 383
Boureanu, Ioana, 1141
Bourgne, Gauvain, 1223
Boutilier, Craig, 737
Bowring, Emma, 1193
Brafman, Ronen, 1265
Brambilla, Manuele, 139
Brandl, Florian, 763
Brandts, Jordi, 1189
Bratman, Jeshua, 407
Brill, Markus, 585
Brito, Ismel, 1263
Brom, Cyril, 1469, 1477
Bromuri, Stefano, 1487
Brown, Matthew, 863
Brunskill, Emma, 1385
Budrene, Elena, 1337
Bügler, Max, 1475
Burnett, Chris, 1359

Caillou, Philippe, 1353
Caire, Giovanni, 1435
Caminada, Martin, 493
Campano, Sabrina, 1191
Čáp, Michal, 1473
Carnevale, Peter, 55
Castelfranchi, Cristiano, 1241
Cataldi, Mario, 1185
Cavallo, Ruggiero, 677
Cavedon, Lawrence, 1081, 1187
Ceppi, Sofia, 1323
Cerquides, Jesus, 1275
Chaganty, Arun Tejasvi, 391
Chakraborty, Nilanjan, 1161

Chalkiadakis, Georgios, 417, 779, 787, 1165
Chang, Yu-Han, 45, 1295, 1505
Chatterjee, Sreerupa, 1333
Chella, Antonio, 1065
Chen Hui, Ong, 29
Chen, Inn-Tung, 1277
Chen, Jianing, 163
Chen, Jie, 105
Chen, Xiaoping, 1215
Chen, Yiling, 889
Chen, Yin, 1437
Cheng, Shih-Fen, 1227
Chhabra, Meenal, 1321
Chi, Luyan, 1295
Chien, Steve, 105
Chli, Maria, 1285
Claes, Daniel, 147, 1495
Cobo, Luis C., 483
Cohen, Robin, 1363
Colby, Mitchell, 425
Colombo Tosatto, Silvano, 1023
Conte, Rosaria, 1189
Corapi, Domenico, 1369
Corruble, Vincent, 1191
Cossentino, Massimo, 1065
Cramer, Henriette, 1197
Crandall, Jacob, 399
Cranefield, Stephen, 1491
Criado, Natalia, 1419
Croitoru, Madalina, 1249

Dam, Hoa, 1433
Damiano, Rossana, 1185
Dannenberg, Roger, 205
Das, Sanmay, 1291, 1321
Dasgupta, Prithviraj, 121
Dastani, Mehdi, 331, 1057, 1133, 1373
d'Avila Garcez, Artur, 1023
Dayama, Pankaj, 703
De Giacomo, Giuseppe, 1031
de Jonge, Dave, 1415
de la Hoz, Enrique, 1259
de Lucena, Carlos, 1225
de Melo, Celso, 55
de Sevin, Etienne, 1191
De Vos, Marina, 1369
de Weerd, Harmen, 1195
del Val, Elena, 1355, 1429
Delle Fave, Francesco Maria, 289, 1467
Demiris, Yiannis, 1207
Derbinsky, Nate, 1387
desJardins, Marie, 315
Devlin, Sam, 433
Di Caro, Gianni, 1205, 1503
Di Loreto, Ines, 1449
Dickerson, John, 711
Dignum, Frank, 1181
Dignum, Virginia, 1183, 1371, 1421
Dimas, Joana, 1175
Dimopoulos, Yannis, 1413
DiRenzo, Joseph, 13
Dobson, Andrew, 247
Dolan, John, 105, 1213
Dorigo, Marco, 97, 139
Doshi, Prashant, 1039, 1243, 1507
Dovgan, Erik, 1485
Driessen, Kurt, 383
Ducatelle, Frederick, 1205
Dunne, Paul, 939
Duong, Quang, 441
Durfee, Ed, 323, 1277

Eck, Adam, 1221
Elidrisi, Mohamed, 1289
Elkind, Edith, 627, 787
Endriss, Ulle, 635
Enz, Sibylle, 1197
Epstein, Daniel, 1389
Erdélyi, Gábor, 627
Ermon, Stefano, 965
Esteva, Marc, 1483

Fabregues, Angela, 1481
Falcone, Rino, 1241
Faliszewski, Piotr, 577
Faltings, Boi, 1273
Fang, Fei, 1299
Fang, Hui, 1365
Farinelli, Alessandro, 417, 1461
Faus, Jaume, 1499
Feige, Uriel, 897
Feldman, Michal, 771, 897
Fernandez, Alberto, 1429
Fink, Andreas, 1417
Fischer, Felix, 585
Fischer, Klaus, 1479
Foss, Bjarne, 1169
Fragiadakis, Daniel, 1327
Frazier, Spencer, 1505
Fredericks, Gary, 1255
French, Tim, 1091
Fridman, Natalie, 1343
Frieder, Asaf, 1281
Friedman, Michal, 1267
Fu, Tuanjie, 1331
Fuentes, Lidia, 1427

Gal, Ya'akov (Kobi), 451
Galstyan, Aram, 1391
Gambardella, Luca, 1205, 1503
Gams, Matjaž, 1485

Ganesan, Vijayalakshmi, 1403
Ganzfried, Sam, 871
Gao, Debin, 29
García-Fornes, Ana, 1377, 1379
Gatti, Nicola, 813, 1323, 1325
Gauci, Melvin, 163
Gaur, Prateek, 391
Gelfand, Michele, 451
Gemrot, Jakub, 1469
Genovese, Valerio, 1023
Genter, Katie, 1251
Gerding, Enrico, 661, 669, 1323
Gerrior, Matthew, 1321
Ghanem, Amer, 551
Ghorbani, Amineh, 1421
Ghose, Aditya, 1433
Ghosh, Siddhartha, 1471
Gil-Quijano, Javier, 1353
Gini, Maria, 1211, 1289
Giret, Adriana, 1493
Giusti, Alessandro, 1503
Gleizes, Marie-Pierre, 1065
Goel, Sharad, 1319
Goldman, Claudia, 459
Gomes, Ana Sofia, 1425
Gomes, Carla, 965
Gonçalves, Ricardo, 1423, 1425
Goodie, Adam, 1243
Goranko, Valentin, 1123
Gordon, Geoff, 1227
Gotta, Danilo, 1435
Governatori, Guido, 1375
Graepel, Thore, 535
Gratch, Jonathan, 55, 63
Grau, Ricardo, 1401
Griffiths, Nathan, 1367
Grimaldo, Francisco, 1357, 1499
Groß, Roderich, 163
Großekathöfer, Ulf, 1177
Grossi, Davide, 805
Grubshtein, Alon, 1267
Grześ, Marek, 1237
Guo, Mingyu, 745
Gutierrez, Patricia, 273, 1263
Gutman, Avital, 719

Hacker, Severin, 467
Hafizoğlu, Feyza, 1349
Haghpanah, Yasaman, 315
Haim, Galit, 451
Han , The Anh, 559
Hanna, Nader, 79, 1463
Harland, James, 1443
Harrenstein, Paul, 585, 1311
Hasegawa, Takato, 795
Hashimoto, Naoyuki, 795
Hayes, Timothy, 21, 1455
Hazon, Noam, 879
Hendrix, Philip, 1337
Hennes, Daniel, 147, 1475, 1495
Hermann, Thomas, 1177
Hernández, Carlos, 997
Hervouet, Fabien, 1449
Hickmott, Sarah, 1163
Hoang, Trong Nghia, 155, 1233
Hoey, Jesse, 1237
Horvitz, Eric, 467, 889, 1329
Hossain, S. G. M. , 121
Hrstka, Ondřej, 37
Hsu, Jane Yung-jen, 1439
Hsu, Wynne, 215
Hu, Cuiyun, 1437
Huang, Lixing, 63

Iliev, Petar, 1115
Inoue, Katsumi, 1223
Iocchi, Luca, 1203
Isbell Jr., Charles L., 483
Ishowo-Oloko, Fatimah, 1167
Itoh, Hidenori, 1159
Iwasaki, Atsushi, 753, 795, 1305, 1327

Jacobson, Michael J., 79, 1463
Jain, Manish, 905
Jain, Shaili, 677
Jakob, Michal, 37, 1501
Jamroga, Wojciech, 1123, 1405
Jazizadeh, Farrokh, 21, 1455
Jeanpierre, Laurent, 1209
Jennings, Nick, 223, 231, 289, 417, 543, 669, 779, 1165, 1167, 1467, 1471
Jiang, Albert Xin, 1299, 1501
Jiang, Jie, 1371
Jiang, Siwei, 299
Jiang, Yichuan, 1331
Jih, Wan-rong, 1439
Joe, YongJoon, 1305
John, Richard, 1297
Johnson, Matthew, 1501
Jones, Andrew, 1141
Jonker, Catholijn, 1183
Julian, Vicente, 1379, 1493
Junges, Robert, 1335

Kaelbling, Leslie, 973
Kaisers, Michael, 947, 1393
Kalia, Anup, 1489
Kaluža, Boštjan, 955, 1485
Kalyanakrishnan, Shivaram, 129
Kamar, Ece, 467, 1329
Kaminka, Gal, 113, 955, 1343

Kandori, Michihiro, 1305
Kang, Sin-Hwa, 63
Kang, Yi-Lin, 1465
Kankanhalli, Mohan, 155
Kannan, Balajee, 1213
Kantor, George, 1213
Karnik, Aditya, 703
Kasneci, Gjergji, 535
Kato, Shohei, 1159
Kavulya, Geoffrey, 21, 1455
Kearns, Michael, 441
Keidar, Matan, 113
Kelaiah, Iwan, 79, 1463
Kerr, Reid, 1363
Keysermann, Matthias, 1197
Kianercy, Ardeshir, 1391
Kiekintveld, Christopher, 863
Kim, Eunkyung, 1295
Kim, Yoonheui, 1279
Kimmel, Andrew, 247
Klein, Laura, 21, 1455
Klein, Mark, 1259
Klenk, Matthew, 989
Klügl, Franziska, 1335
Klusch, Matthias, 1479
Knobbout, Max, 331
Knorr, Matthias, 1425
Knox, W. Bradley, 475
Koenig, Sven, 997
Kok, Eric, 1411
Kollar, Thomas, 1217
Komenda, Antonín, 1239, 1473, 1501
Kopp, Stefan, 1177
Kosinski, Michal, 535
Koster, Andrew, 517
Kot, Alex C., 1361
Kota, Ramachandra, 1165
Kraemer, Landon, 1441
Krampf, Johannes, 1487
Kraus, Sarit, 451, 459, 1281, 1297
Kriegel, Michael, 1457
Kudenko, Daniel, 433
Kuter, Ugur, 981, 989, 1201
Kwak, Jun-young, 21, 1455

Lagoudakis, Michail, 171
Lai, Darong, 1331
Laird, John, 1387
Lang, Fabian, 1417
Lang, Jérôme, 577, 585, 1247
Langford, John, 1317
Lau, Hoong Chuin, 257, 1227
Lau, Qiangfeng Peter, 215
Lazaric, Alessandro, 1325
Lee, Mong Li, 215
Leite, João, 1425
Leonetti, Matteo, 1203
Lespérance, Yves, 1031
Lesser, Victor, 1279
Letchford, Joshua, 1303
Lev, Omer, 611
Lewis, Bennie, 1219
Lewis, Michael, 1161
Lewis, Richard, 407
Li, Justin, 1387
Li, Minyi, 525
Liang, Han, 1261
Lin, Raz, 1281
Linkov, Igor, 1337
Lisý, Viliam, 1301, 1473
Littman, Michael, 947
Liu, Fan, 1459
Liu, Siyuan, 1361
Logan, Brian, 1057, 1309
Lombardo, Vincenzo, 1185
Lomuscio, Alessio, 1141
Lopez Carmona, Miguel Angel, 1259
Lopez-Mobilia, Adrian, 129
Lopez-Sanchez, Maite, 1483
Lorini, Emiliano, 1133
Low, Kian Hsiang, 105, 155, 1233
Luck, Michael, 1225
Luke, Sean, 197
Luo, Jian, 1015
Luštrek, Mitja, 1485
Lyle, Jeremy, 1441

MacAlpine, Patrick, 129
Machado , André, 1389
Magee, Liam, 1163
Maghami, Mahsa, 687
Magnenat Thalmann, Nadia, 1365
Mahdian, Mohammad, 1319
Maheswaran, Rajiv, 21, 45, 1295, 1297, 1299, 1455, 1505
Maleki, Sasan, 1471
Mamidi, Sunil, 45
Manenti, Lorenza, 1341
Mao, Hua, 1015
Mao, Xinjun, 1437
Marecki, Janusz, 821, 1235
Markakis, Vangelis, 779
Marsa-Maestre, Ivan, 1259
Marsella, Stacy, 1193
Martinho, Carlos, 1175
Mathews, Nithin, 97
Matignon, Laëtitia, 1209
Mattei, Nicholas, 1407
Maudet, Nicolas, 1223, 1313
Maule, Ben, 13
Mauri, Alessia, 1339

McClean, Sally, 417
Meeussen, Wim, 147, 1495
Mehrotra, Siddharth, 1161
Meir, Reshef, 771
Meisels, Amnon, 1267
Meneguzzi, Felipe, 1161
Meseguer, Pedro, 273, 1263
Męski, Artur, 1447
Meyer, Garrett, 13
Meyer, John-Jules, 1411
Miao, Chunyan, 1361, 1465
Michalak, Tomasz, 239
Migeon, Frédéric, 1065
MikoŁaj, Mikolaj, 493
Miles, Simon, 1225
Miller, Sam, 281
Minai, Ali, 551
Moler, Zbyněk, 1501
Molineaux, Matthew, 989
Moraitis, Pavlos, 1413
Morel, Benoit, 1337
Morency, Louis-Phillippe, 63
Mouaddib, Abdel-Illah, 1209
Mouri, Takayuki, 753
Muise, Christian, 1031
Mutoh, Atsuko, 1159

Nagi, Jawad, 1503
Nahum, Yinon, 1291
Nakisaee, Ali, 1497
Nanjanath, Maitreyi, 1211
Narahari, Yadati, 703
Narayanan, Ajit, 1459
Natarajan, Prabhu, 155
Nau, Dana, 981, 1201
Nelson, Carl, 121
Newnan, Alex, 1505
Newstead, Anne, 79, 1463
Ng, Wing Lon, 653
Nguyen, Duc Thien, 257
Nguyen, Nhan-Tam, 1287
Nguyen, Trung Thanh, 1287
Ning, Yu, 1295
Nisan, Noam, 719
Nissim, Raz, 1265
Noriega, Pablo, 1419, 1421
Norman, Timothy, 1409
Novák, Peter, 1239, 1473
Nowé, Ann, 1401
Nunes, Ernesto, 1211
Nunes, Ingrid, 1225

Obara, Ichiro, 1305
Ochs, Magalie, 87
Ogawa, Yuki, 1347
Oh, Jean, 1161
Ohta, Naoki, 795
Okada, Isamu, 1347
Okamoto, Steven, 879
Oliehoek, Frans, 973, 1229
Onaindia, Eva, 509
Ong, Yew-Soon, 299, 1465
Ordóñez, Fernando, 847, 863
Oren, Nir, 1359
Ossowski, Sascha, 1173
Othman, Abraham, 645
Ottens, Brammert, 1273

Padget, Julian, 1369
Padgham, Lin, 1049, 1081, 1163, 1187
Pajares Ferrando, Sergio, 509
Palit, Imon, 653
Pan, Yinghui, 1015
Panozzo, Fabio, 813
Paolucci, Mario, 1357
Paraschos, Alexandros, 171
Pardo, Pere, 1231
Parkes, David, 889
Parr, Gerard, 417
Pěchouček, Michal, 37, 905, 1239, 1301, 1473, 1501
Pedersen, Sindre, 1169
Pelachaud, Catherine, 87, 1179
Peled, Hilla, 265
Peña, Jorge, 1175
Penczek, Wojciech, 1447
Pennock, David, 1317, 1319
Penya-Alba, Toni, 1275
Pereira , Luís Moniz, 559
Perez, Victor, 1493
Pham, Manh Tung, 1257
Phelps, Steve, 653
Pinciroli, Carlo, 139
Pini, Maria Silvia, 1313, 1407
Pita, James, 1297
Piunti, Michele, 1241
Pizzo, Antonio, 1185
Popelová, Markéta, 1477
Porte, John, 79, 1463
Prada, Rui, 1175
Prakken, Henry, 1411
Prepin, Ken, 1179
Price, Michael J., 163
Procaccia, Ariel, 711
Proper, Scott, 1397
Pryymak, Oleksandr, 543
Pyrga, Evangelia, 1311

Qu, Xia, 1243

Rabinovich, Zinovi, 459
Raboin, Eric, 1201
Rahwan, Iyad, 493, 1167

Rahwan, Talal, 239
Raimundo, Guilherme, 1175
Rajagopal, Karthik, 1299
Ramamoorthy, Subramanian, 349, 1203
Ramchurn, Sarvapali, 223, 281
Ranathunga, Surangika, 1491
Ranjbar-Sahraei, Bijan, 1497
Rao, Karun, 375
Ravindran, Balaraman, 391
Read, Stephen, 55
Rebollo, Miguel, 1355, 1429
Reeves, Daniel, 1319
Reijngoud, Annemieke, 635
Ribeiro, Luís Landeiro, 1175
Richards, Deborah, 79, 1463
Riedl, Mark, 71
Robu, Valentin, 669, 1165
Rodríguez, Abdel, 1401
Rodriguez, Inmaculada , 1483
Rodriguez, Rosa Maria, 1493
Rodriguez-Aguilar, Juan Antonio, 1275
Rogers, Alex, 281, 289, 417, 543, 661, 1165, 1461, 1467, 1471
Roos, Magnus, 1287
Rose, Harry, 661
Rosenschein, Jeffrey, 611, 1315
Rossi, Francesca, 1313, 1407
Rothe, Jörg, 577, 1287
Rovatsos, Michael, 307
Rudolph, Sebastian, 1249
Ruff, Alexander, 1333
Rui, Zaojie, 1331
Ruiz, N., 1493
Russo, Alessandra, 1369

Sá, Samy, 501
Sabater-Mir, Jordi, 517, 1189
Sabouret, Nicolas, 1191
Sadat, Seyed Abbas, 1199
Sadrzadeh, Mehrnoosh, 1231
Saffidine, Abdallah, 1247
Samadi, Mehdi, 1217
Sandholm, Tuomas, 645, 711, 729, 871
Santos, Francisco C., 559
Santos, Pedro A., 1175
Sardina, Sebastian, 1049, 1081
Sarne, David, 1291, 1293, 1445
Satoh, Ken, 1369
Sattar, Abdul, 1375
Scerri, Adrian, 1213
Scerri, Paul, 1213, 1261, 1269
Scheidler, Alexander, 97
Scheutz, Matthias, 189
Schjølberg, Ingrid, 1169
Schorlemmer, Marco, 517
Schumacher, Michael, 1487
Schumann, René, 1487
Seedig, Hans Georg, 585
Segal, Richard, 821
Seidita, Valeria, 1065
Selman, Bart, 965
Sen, Sandip, 1333, 1349
Şensoy, Murat, 1365
Seow, Kiam Tian, 1257
Serrano, Emilio, 307, 1377
Service, Travis, 569, 593
Shankar, Kumar Shaurya, 1213
Shapiro, Steven, 1081
Shehory, Onn, 1291
Shieh, Eric, 13
Shivashankar, Vikas, 981
Sidner, Candy, 63
Sierra, Carles, 1415, 1481
Singh, Munindar, 1073, 1149, 1489
Singh, Satinder, 407, 441, 1277
Slavkovik, Marija, 1403, 1405
Sleight, Jason, 323
Slota, Martin, 1425
Slusallek, Philipp, 1479
Smith, Stephen, 1271
Sofy, Nadav, 1445
Soh, Leen-Kiat, 1221
Song, Zhao, 1199
Sonu, Ekhlas, 1039, 1507
Sørensen, Troels, 829
Sorg, Jonathan, 407
Sousa, Sergio, 1403
Spaan, Matthijs, 1229
Spanoudakis, Nikolaos, 171
Sridharan, Mohan, 181
Stein, Sebastian, 231, 669
Ştiurcă, Nicolae, 129
Stone, Peter, 129, 341, 357, 475, 1251
Stranders, Ruben, 289
Stranieri, Alessandro, 97
Subagdja, Budhitama, 1007, 1465
Such, José, 1377
Suetsugu, Katsuya, 1159
Sukthankar, Gita, 687, 1219
Sullivan, Keith, 197
Sutanto, Danny, 1253, 1431
Sycara, Katia, 879, 1161, 1269, 1409
Szczepański, Piotr, 239
Szreter, Maciej, 1447

Tambe, Milind, 13, 21, 847, 855, 863, 905, 955, 1193, 1297, 1299, 1307, 1455, 1501
Tan, Ah-Hwee, 1007, 1465
Tan, Yao-Hua, 1371
Tan, Yuan-Sin, 1007

Tang, Pingzhong, 729
Tay, Junyun, 205
Taylor, Charlotte, 79, 1463
Taylor, Matthew, 383, 1383
Taylor, Meredith, 79, 1463
Teacy, Luke, 417
Telang, Pankaj, 1073, 1489
Tennenholtz, Moshe, 771, 897
Teow, Loo-Nin, 1007
Tesauro, Gerry, 821
Tettamanzi, Andrea, 1339
Thakur, Subhasis, 1375
Thangarajah, John, 1049, 1081, 1187, 1443
Theng, Yin-Leng, 1361
Thom, James, 1163
Thomaz, Andrea, 483
Thompson, David, 105
Tittle, James, 1161
Tjønnås, Johannes, 1169
Todo, Taiki, 753
Tomaszewski, Christopher, 1213
Tomek, Jakub, 1477
Toniolo, Alice, 1409
Torrey, Lisa, 1383
Tran-Thanh, Long, 289
Trescak, Tomas, 1483
Trodd, Luke, 1443
Troquard, Nicolas, 1245
Trovò, Francesco, 1325
Troyan, Peter, 1327
Tsai, Jason, 1193
Tsao, Tiffany Yi-Ting, 1439
Tsimhoni, Omer, 459
Tumer, Kagan, 425, 1397
Turrini, Paolo, 805
Tuyls, Karl, 147, 383, 1393, 1475, 1495

Ueda, Suguru, 795, 1327
Ufimtsev, Vladimir, 121
Uras, Tansel, 997
Urieli, Daniel, 129

Valada, Abhinav, 1213
van der Hoek, Wiebe, 1115
van der Schaar , Mihaela, 1283
van der Torre, Leendert, 1023, 1373, 1403
van der Zwaan, Janneke, 1183
van Ditmarsch, Hans, 1091, 1247
Van-Gael, Jurgen, 535
van Oijen, Joost, 1181
Vaněk, Ondřej, 37, 905
Varakantham, Pradeep, 21, 29, 1227, 1235, 1269, 1455
Vargas, Patricia, 1197
Vasirani, Matteo, 1173, 1429
Vaughan, Richard T., 1199
Vedanarayanan, Srinivasa, 551
Velagapudi, Prasanna, 1213, 1269
Velázquez-Quesada, Fernando R., 1091
Veloso, Manuela, 205, 1217
Venable, Kristen Brent, 1313, 1407
Venanzi, Matteo, 1241
Verbrugge, Rineke, 1195
Verheij, Bart, 1195
Verwer, Sicco, 1399
Vidal, José, 1255
Villatoro, Daniel, 1189
Vinyals, Meritxell, 1275, 1461
Vizzari, Giuseppe, 1341
Vladimirsky, Alexander, 965
Vo, Bao, 525
Voice, Thomas, 223
Vorobeychik, Yevgeniy, 1303, 1307
Vrancx, Peter, 1401
Vreeswijk, Gerard, 1411
Vu, Victor, 129
Vytelingum, Perukrishnen, 1167

Waizman, Gennady, 1453
Wallace, Iain, 1457
Walsh, Toby, 603
Wang, Wenwen, 1007
Warwas, Stefan, 1479
Waugh, Kevin, 871
Weiss, Gerhard, 383, 1497
Wellman, Michael, 441, 931
Whiteson, Shimon, 375
Wiedenbeck, Bryce, 931
Williamson, Simon, 29, 231
Winsper, Michael, 1285
Witwicki, Stefan, 973, 1277
Wöhler, Nils-Christian, 1177
Wong, Wilson, 1187
Wood, Wendy, 21, 1455
Wooldridge, Michael, 939, 1115
Woźna-Szcześniak, Bozena, 1447
Wu, Feng, 1215
Wunder, Michael, 947
Wyner, Adam, 1171

Xia, Guangyu, 205
Xia, Lirong, 603
Xie, Xiao-Feng, 1271
Xu, Jie, 1283
Xu, Yang, 1261

Yamamoto, Hitoshi, 1347
Yang, Rong, 13, 847, 1297, 1299
Yaros, John Robert, 947
Ye, Dayong, 1253, 1431
Yeoh, William, 257, 1227, 1269
Yin, Zhengyu, 855, 905, 1501
Yokoo, Makoto, 753, 795, 1305, 1327

Yorke-Smith, Neil, 1373
Yu, Hong, 71

Zame, William, 1283
Zappala, Julian, 1309
Zbrzezny, Andrzej, 1447
Zeng, Yifeng, 1015
Zhang, Haoqi, 889
Zhang, Jie, 299, 1365
Zhang, Minjie, 1253, 1431
Zhang, Shiqi, 181
Zhang, Yingqian, 1399
Zhou, Huiping, 1437
Zhu, Linglong, 1261
Zick, Yair, 787
Zilka, Avishay, 1343
Zivan, Roie, 265, 1267
Zoll, Carsten, 1197
Zuckerman, Michael, 1315